# An Area-Efficient and Low-Power Multirate Decoder for Quasi-Cyclic Low-Density Parity-Check Codes

Bo Xiang, Rui Shen, An Pan, Dan Bao, and Xiaoyang Zeng, *Member, IEEE*

***Abstract*—The quasi-cyclic low-density parity-check (QC-LDPC) codes are widely applied in digital broadcast and communication systems. However, the decoders are still difficult to be put into practice due to their large area and high power, especially in the wireless mobile devices. This paper presents an improved all-purpose multirate iterative decoder architecture for QC-LDPC codes, which can largely reduce their area and power. The architecture implements the normalized min-sum algorithm, rearranges the original two-phase message-passing flow, and adopts an efficient quantization method for the second minimum absolute values, an optimized storing scheme for the position indexes and signs, and an elaborate clock gating technique for substantive memories and registers. It is also configurable for any regular and irregular QC-LDPC codes, and can be easily tuned up to different code rates and code word lengths. The chip is fabricated in an SMIC 0.18-$\mu$m six-metal-layer standard CMOS technology. It attains a throughput of 104.5 Mb/s, and dissipates an average power of 486 mW at 125 MHz, and 15 decoding iterations. The core area is only 9.76 mm$^2$. The chip has been applied into the China digital terrestrial/television multimedia broadcasting system.**

***Index Terms*—Digital terrestrial/television multimedia broadcasting (DTMB) system, iterative decoder architecture, min-sum-with-normalization-factor algorithm, quasi-cyclic low-density parity-check (QC-LDPC) codes, VLSI implementation.**

## I. Introduction

FORWARD error correction (FEC) codes have been playing an important role in the digital broadcast and communication systems. Among all kinds of FEC codes, the low-density parity-check (LDPC) codes, which were first invented by Gallager [1], have become more and more popular nowadays, since it was proved by MacKay and Neal [2] that the performance of the LDPC codes was remarkable and even much closer to Shannon limit than that of other codes.

The LDPC codes, which are defined by a sparse parity-check matrix, are one kind of linear block codes, and can be efficiently presented by the bipartite Tanner graph [3]. There are two primary types of parity-check matrices: the pseudorandom matrix [1] and the quasi-cyclic matrix [4], [5]. The latter one, whose encoding complexity is directly proportional to the code word length, is widely applied in the systems such as DVB-S2, 802.11n, 802.16e, China digital terrestrial/television multimedia broadcasting (DTMB) system, magnetic storage and recording system, and maybe next generation wireless mobile communication system.

A structured sparse parity-check matrix $\boldsymbol{H}$ defines the quasi-cyclic low-density parity-check (QC-LDPC) codes $(M,\ N,\ b,\ t)$. The matrix $\boldsymbol{H}$ is composed of $M \times N$ square submatrices. Each submatrix is a $b \times b$ either all zero or circulant permutated square matrix. The total number of nonzero submatrices is $t$. Therefore, the QC-LDPC codes can be compactly identified by the base matrix, which only stores the position indexes and offsets of all nonzero submatrices in $\boldsymbol{H}$. The irregular multirate QC-LDPC codes are employed in the DTMB system. The parameters $(M,\ N,\ b,\ t)$ are (35, 59, 127, 275), (23, 59, 127, 296), and (11, 59, 127, 294) for code rates 2/5, 3/5, and 4/5, respectively. The matrix $\boldsymbol{H}$ and its partial Tanner graph for code rate 2/5 is shown in Fig. 1, where $W_r$ is the row weight and $W_c$ is the column weight.

It can perform very well with outstanding error-correcting performance for LDPC decoders using iterative decoding algorithms. The original two-phase message-passing (TPMP) algorithm [1] has better performance with higher complexity. Contrarily, the min-sum algorithm [6] has lower complexity with worse performance. For the tradeoff between performance and complexity, some variations of min-sum algorithm come forth. One is the min-sum-plus-correction-factor algorithm [7], and the other is the min-sum-with-normalization-factor algorithm [8], [9]. The former one, marked with min-sum $c$, just saves the memory bits for lookup tables. However, the latter one, marked with min-sum $\alpha$, is much more realizable because it can largely reduce the memory bits for extrinsic messages. In addition, the turbo-decoding message-passing (TDMP) algorithm [10] has been applied in LDPC decoders with faster convergence speed.

Based on the iterative decoding algorithms, many QC-LDPC decoder architectures have been implemented for different applications. In general, they would fall into: 1) the architectures with better performance; 2) the architectures with higher throughputs; 3) the architectures with smaller area; 4) the architectures with lower power; and 5) the architectures with more flexible configurability. The decoder architectures [10], [11], adopting the TDMP algorithm, have faster convergence speed and dispense with extra memories for intrinsic messages. However, they still need good-sized memory bits and have higher computing complexity. The decoder architectures [12], [13], applied into the magnetic storage and recording systems, can attain higher throughputs for both phases are interleaved

Manuscript received September 20, 2008; revised January 31, 2009. First published September 01, 2009; current version published September 24, 2010. This work was supported by the Science and Technology Commission of Shanghai Municipality under Grant 77062001.

The authors are with the State Key Laboratory of Application-Specific IC and System, Department of Microelectronics, Fudan University, Shanghai 201203, China (e-mail: bxiang@fudan.edu.cn; rshen@fudan.edu.cn; 062052022@fudan.edu.cn; 062052032@fudan.edu.cn).

Digital Object Identifier 10.1109/TVLSI.2009.2025169

with each other. Nevertheless, they are not area efficient for longer code word lengths and lower code-rates. Additionally, they are suffering from a little more performance loss and lack of flexibility. The decoder architectures [14], [15], introducing the overlapped message-passing methods, can increase the throughputs to some extent. However, they rest heavily with the structures of matrix $\boldsymbol{H}$. The decoder architectures [16], [17], which are called vertical shuffle schedule (VSS), can speed up convergence and increase throughputs. Nevertheless, they need so many check-node updating units when the number of rows in $\boldsymbol{H}$ gets larger that the area and power may not be acceptable for the mobile terminals. The traditional partially parallel decoder architectures [14], [15], [18]–[21] dispense with permutation network. However, they are not configurable and not area efficient for layout floor planning because of large numbers of memories.

In this paper, based on the min-sum-$\alpha$ algorithm [8], following the idea of repartitioning both updating phases [22], a configurable area-efficient and low-power multirate decoder for QC-LDPC codes has been implemented. The decoder can be applied into set-top boxes and mobile terminals, which are critical for the area and power. The paper is organized as follows. Section I is introduction. Section II briefly reviews the logarithmic domain TPMP and min-sum-$\alpha$ algorithms, and advances some modifications that aim at lowering the cost of VLSI implementation. It also gives the performance curves. Section III presents the proposed decoder architecture, and describes the techniques to decrease the area and power in detail. The implementation results are followed in Section IV. Finally, Section V concludes the paper.

## II. Rearranged Algorithm and Quantization

The Gallager's TPMP algorithm [1] propagates the extrinsic messages between the check nodes and the variable nodes in multiple rounds of iterations. The *a posteriori* probabilities can converge to either 1 or 0 after some decoding iterations.

### A. Logarithmic Domain TPMP Algorithm

Let $Lr_{c\to v}^{(k)}$ be the extrinsic logarithm likelihood ratio (LLR) from check node $c$ to variable node $v$ in the $k$th iteration; $Lq_{v\to c}^{(k)}$, the extrinsic LLR from variable node $v$ to check node $c$ in the $k$th iteration; $Li_v$, the channel intrinsic LLR of variable node $v$; $LQ_v^{(k)}$, the posterior LLR of variable node $v$ after the $k$th iteration; $N(c)$, the set of check nodes connected with check node $c$; $N(c)\backslash v$, the set of check nodes connected with check node $c$ excluding variable node $v$; $M(v)$, the set of check nodes connected with variable node $v$; and $M(v)\backslash c$, the set of variable nodes connected with variable node $v$ excluding check node $c$. The iteration index $k$ is an integer from 1 to $N_{\text{iter}}$., where $N_{\text{iter}}$. is the maximum iteration number.

The check node updating operation can be expressed as follows:

$$Lr_{c\to v}^{(k)} = \prod_{n\in N(c)\backslash v} \operatorname{sgn}(Lq_{n\to c}^{(k-1)}) \times \phi\left(\sum_{n\in N(c)\backslash v} \phi(\operatorname{abs}(Lq_{n\to c}^{(k-1)}))\right). \tag{1}$$

Functions $\operatorname{sgn}(x)$ and $\operatorname{abs}(x)$ in (1) denote the sign and absolute value of $x$, respectively, and $\phi(x)$ is given by

$$\phi(x) = \phi^{-1}(x) = \log\left(\frac{(e^x+1)}{(e^x-1)}\right). \tag{2}$$

The variable node updating operation can be expressed as follows:

$$Lq_{v\to c}^{(k)} = Li_v + \sum_{m\in M(v)\backslash c} Lr_{m\to v}^{(k-1)}. \tag{3}$$

The variable nodes are decoded according to the following equations:

$$LQ_v^{(k)} = Li_v + \sum_{m\in M(v)} Lr_{m\to v}^{(k-1)}$$
$$\hat{c}_v = \begin{cases} 1, & \text{if } LQ_v^{(k)} \geq 0 \\ 0, & \text{if } LQ_v^{(k)} < 0. \end{cases} \tag{4}$$

The iterative decoding process terminates when $\hat{c}H^T = \mathbf{0}$ or $k = N_{\text{iter}}$.

### B. Rearranged Min-Sum-$\alpha$ Algorithm

The min-sum-$\alpha$ algorithm proposed in [8] is simply derived from substituting (5) for (1). Here, $\alpha$ is the normalization factor, which is related with channels and code rates. To obtain the best performance, the factor should vary from one iteration to another, and depends on the SNR values. However, to make the algorithm as simple as possible, the factor is constant for all iterations and all SNR values [9]

$$Lr_{c\to v}^{(k)} = \alpha \times \prod_{n\in N(c)\backslash v} \operatorname{sgn}(Lq_{n\to c}^{(k-1)}) \times \min_{n\in N(c)\backslash v} (\operatorname{abs}(Lq_{n\to c}^{(k-1)})). \tag{5}$$

After simplification by the min-sum-$\alpha$ algorithm, the computation flow can be rearranged [22] as in (6) and (7). For each check node $c$, the block of searcher searches the first and the second minimum absolute values through the extrinsic messages from all variable nodes $n \in N(c)$ to check node $c$. Meanwhile, it stores the position index of the first minimum absolute value and the signs of all extrinsic messages. For variable node $v$, the block of accumulator computes the sum of the channel intrinsic message and the extrinsic messages from all check nodes $m \in M(v)$ to variable node $v$

$$[\,1\text{stmin} \quad 2\text{ndmin} \quad \text{pos} \quad \text{sgn}\,]_c^{(k)} = \underset{n\in N(c)}{\text{Searcher}}(LQ_n^{(k-1)} - Lr_{c\to n}^{(k-1)}) \tag{6}$$

$$LQ_v^{(k)} = \underset{m\in M(v)}{\text{Accumulator}}(Lr_{m\to v}^{(k-1)}) + Li_v. \tag{7}$$

Both variables, $Lr_{c\to n}^{(k-1)}$ in (6) and $Lr_{m\to v}^{(k-1)}$ in (7), are recovered from the following equation:

$$Lr_{c\to v}^{(k-1)} = \alpha \times \text{Recover}([\,1\text{stmin} \quad 2\text{ndmin} \quad \text{pos} \quad \text{sgn}\,]_c^{(k-1)}). \tag{8}$$

The rearranged min-sum-$\alpha$ algorithm imparts three key advantages for VLSI implementation as follows.

1) It can reduce the number of memory bits and the number of memory accesses per iteration in a great measure.

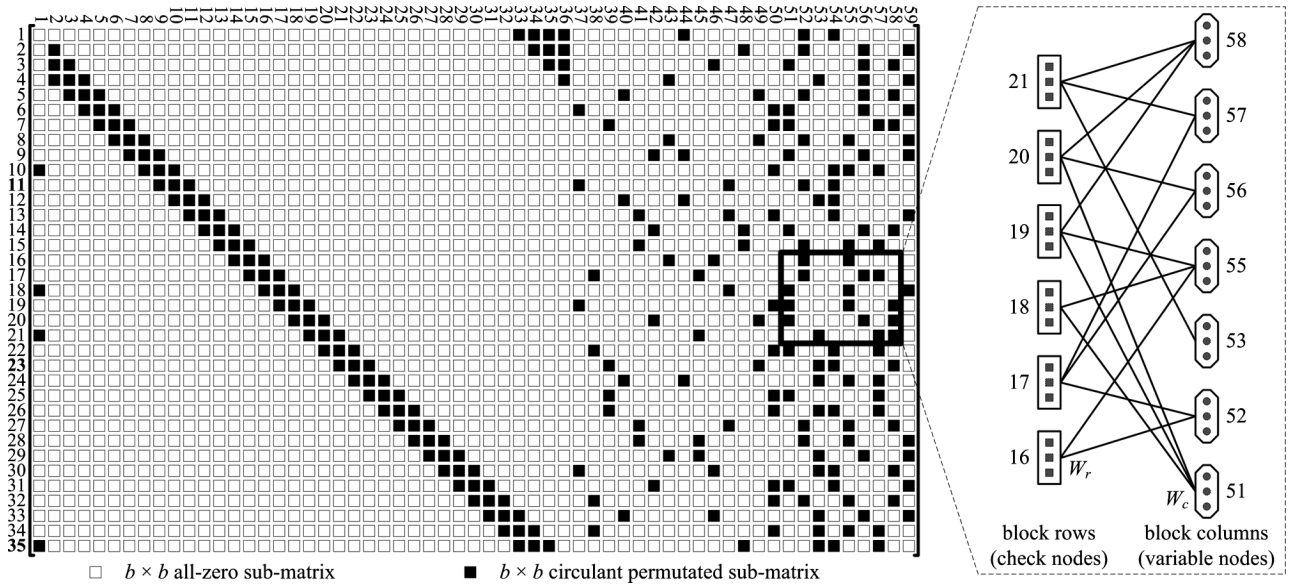



Fig. 1. Irregular QC-LDPC code $(M,\ N,\ b,\ t)$ and its partial Tanner graph in the DTMB system, where $M = 35$, $N = 59$, $b = 127$, and $t = 275$ for code rate 2/5.

2) It is much more flexible to design configurable partially parallel decoder architectures.
3) The check node updating phase and the variable node updating phase can share the same processor if both of them work in turn.

For 1), only the first and the second minimum absolute values are saved in the check node updating phase, and only the sum of each column is saved in the variable node updating phase. In addition, extra memories are required to store the signs of all the extrinsic messages and the position indexes of the first minimum absolute values in each row. Compared with the traditional decoder architectures [14], [15], [18]–[21], the proposed decoder architecture can cut down the number of memory bits by about 59.5%, and the number of memory accesses per iteration by about 68% consequently.

For 2), a novel configurable partially parallel multirate decoder architecture, which is different from the architectures [13], [22], [23], is presented for any regular and irregular QC-LDPC codes. In the proposed architecture, the rearranged two phases work yet in turn, but each row or column is updated serially. To increase the decoder's throughputs, $b$ adjacent rows or columns are concurrently updated. Section IV will discuss the proposed decoder architecture in detail.

For 3), (6) and (7) indicate that the rearranged TPMP process features fine granularity. Subtraction and comparison are the basic arithmetic logical units (ALUs) in the check node updating phase, and addition is the basic ALU in the variable node updating phase. As a result, the hardware calculating resources can be approximately reduced by 40% when two equations are realized by the same processor. With the technique of time-division multiplexing, both phases cannot be processed concurrently. Nevertheless, the decoder's area can be reduced further, and the throughputs can be doubled with the twofold work frequency.

### C. Quantization and Performance Curves

Based on the Monte Carlo simulation and the field-programmable gate array (FPGA)-based field testing, the normalization factor $\alpha$ in the DTMB system equals 0.625, 0.5625, and 0.625 for code rates 2/5, 3/5, and 4/5, respectively. Thereby, the complicated multiplication operation can be realized by the simple shift-and-add operation.

The decoder uses 6-bit integer and 2-bit fraction to quantize the LLRs. An 8-bit quantization gets much better error-correcting performance than 7-, 6-, and 5-bit quantizations. Fig. 2 shows the bit error ratio (BER) curves on the additive white Gaussian noise (AWGN) channel for different code rates when adopting 8-, 7-, 6-, and 5-bit quantizations. In Fig. 2, the left, middle, and right clusters of BER curves are for code rates 2/5, 3/5, and 4/5, respectively.

For high code rate 4/5, the performance curves of different quantizations are nearly the same with one another. They make a much more striking difference when the code rate gets lower. Especially, for code rates 3/5 and 2/5, the performance curves of 5-bit quantization do not drop like the waterfall. They give rise to high error floor [24] when the BER is less than $10^{-4}$. Taking much more complicated channels into account, the decoder uses 8-bit quantization with 2-bit fraction to meet the high performance requirements of high-definition TV (HDTV) systems.

The second minimum absolute values feature the following two statements: 1) they are used only once $(1/W_r)$ in each row in the check node updating phase and 2) simulation results indicate that the mean of differences between the second and the first minimum absolute values is very small. In order to save memory bits, the decoder stores only the first minimum absolute values quantized by 7-bit unsigned digit and the differences quantized by 4-bit unsigned digit. Consequently, the decoder can save memory resources of $35 \times 127 \times 3 = 13\,335$ bits, which are about 5.64% of the total memory bits used in the proposed decoder architecture.

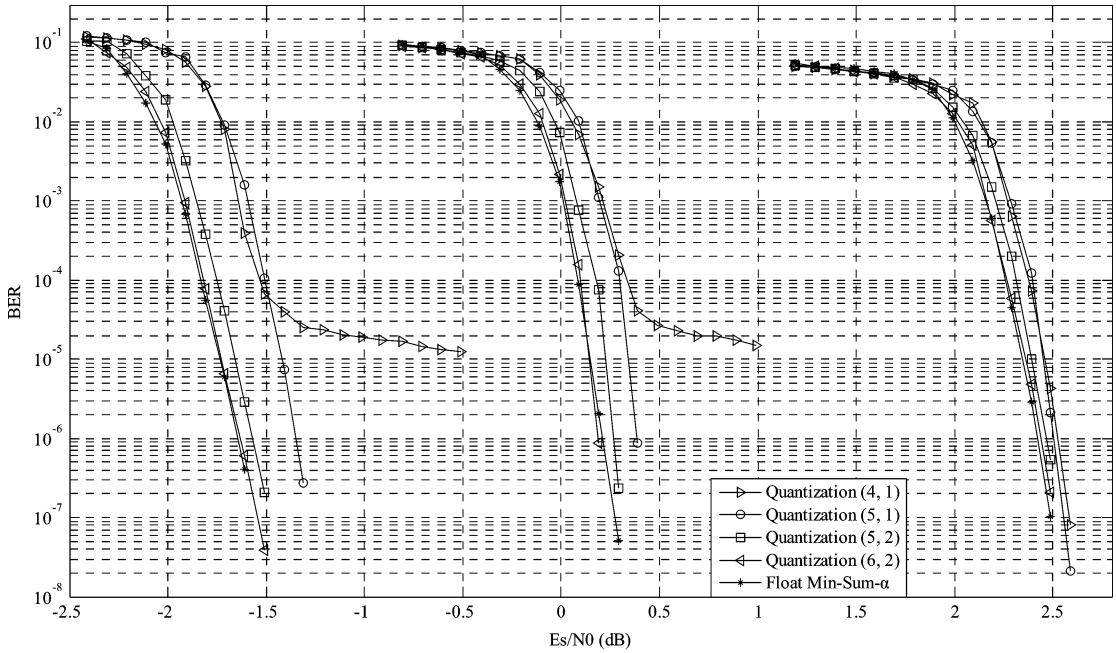



Fig. 2. BER curves of min-sum-$\alpha$ algorithm employing different quantization schemes on the AWGN channel. The left, middle, and right parts are for code rates 2/5, 3/5, and 4/5, respectively, the normalization factor equals 0.625, 0.5625, and 0.625 correspondingly, and the maximum iteration number equals 15.

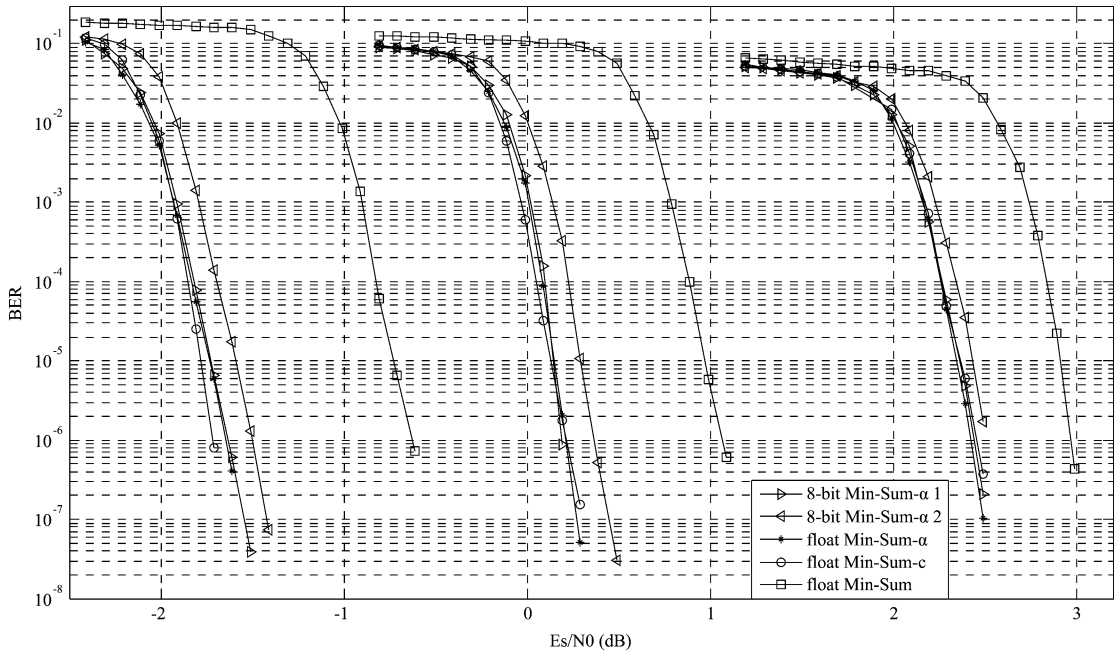



Fig. 3. BER curves of different algorithms on the AWGN channel. The left, middle, and right parts are for code rates 2/5, 3/5, and 4/5, respectively, the normalization factor equals 0.625, 0.5625, and 0.625 correspondingly, and the maximum iteration number equals 15.

If the differences overflow, they would be truncated to the maximum value of 4-bit binary sequence $(1111)_2$. In this case, the 4-bit representation with overflow protection will slow the convergence speed down and result in a little performance loss. Actually, the performance loss heavily depends on the code rate CR . For all the QC-LDPC codes $(M,\ N,\ b,\ t)$, the row weight $W_r$ becomes larger when the code rate CR becomes higher. The expression $1/W_r$, which denotes the total probability of utilizing the second minimum absolute values for all the extrinsic messages, becomes smaller consequently. That is to say, the performance loss becomes smaller when the code rate CR becomes higher. Fig. 3 shows the performance loss for different code rates. Compared with the 7-bit quantization method for all the second minimum absolute values (marked with 8-bit min-sum-$\alpha$ 1), the 4-bit quantization method for all the differences (marked with 8-bit min-sum-$\alpha$ 2) degrades the error-correcting performance only by about 0.15 dB at code rate 2/5 and about 0.08 dB at code rate 4/5.

As shown in Fig. 3, the performance of min-sum-$\alpha$ algorithm is nearly the same with that of min-sum-$c$ algorithm. However,

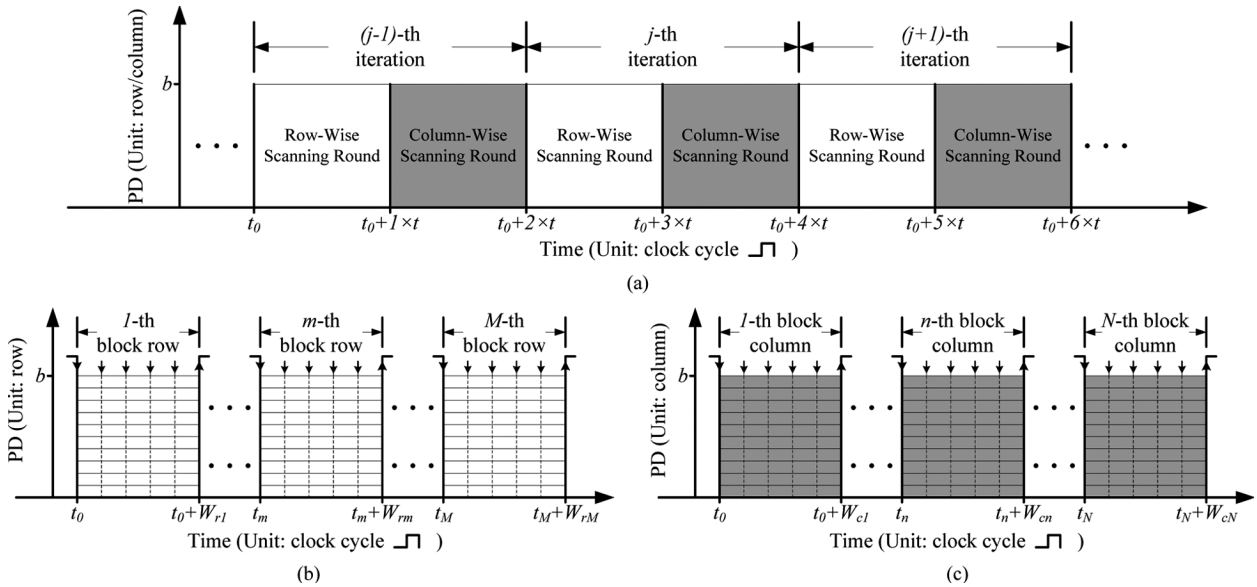



Fig. 4. Time schedule of the proposed decoder architecture for QC-LDPC codes $(M,\ N,\ b,\ t)$: (a) overall time schedule, (b) time schedule of the RWSR, and (c) time schedule of the CWSR. Here, $b = 10, W_{rm} = 5$ for $m = 1, \ldots, M$, and $W_{cn} = 5$ for $n = 1, \ldots, N$.

both of them are about 0.8 dB better than the simplest min-sum algorithm.

## III. Proposed Iterative Decoder Architecture

For the tradeoff between throughputs and area, the partially parallel decoder architectures are often taken into consideration. *Parallel degree (PD)* is a key parameter to illuminate the decoder architectures. In the traditional partially parallel architectures in [14], [15], and [18]–[21] for QC-LDPC codes $(M,\ N,\ b,\ t)$, PD is either the number of block rows $M$ or the number of block columns $N$. In other words, $M$ rows from different block rows are updated simultaneously in the check node updating phase, and $N$ columns from different block columns are updated simultaneously in the variable node updating phase. All rows or columns of each block are updated serially in $b$ clock cycles. In this case, the number of memories is directly proportional to the number of nonzero submatrices $t$ in $\boldsymbol{H}$. These architectures feature such coarse granularity that both phases cannot share the same processor when they work in turn. Thereby, the cost of VLSI implementations is still very high. To reduce the area and power further, we proposed a novel configurable partially parallel multi-rate decoder architecture for any regular and irregular QC-LDPC codes $(M,\ N,\ b,\ t)$.

### A. Timing Schedule

The propagated extrinsic messages between check nodes and variable nodes are recovered from (8). Therefore, the decoder can be realized much more efficiently by changing the value of PD to $b(=\ 127)$. Different from the architectures in [14], [15], and [18]–[21], $b$ rows or $b$ columns of each block are updated concurrently, and the whole updating process is accomplished serially by scanning matrix $\boldsymbol{H}$ in the row-wise order and column-wise order alternately. A whole decoding iteration is composed of a row-wise scanning round (RWSR) operation and a column-wise scanning round (CWSR) operation. The two scanning rounds can be scheduled in two ways. First, both of them are processed concurrently. Second, both are processed serially. The proposed iterative decoder architecture employs the second way to optimize the area and power, and doubles the throughputs with the twofold work frequency.

The timing schedule is demonstrated in Fig. 4, where $W_{rm}$ for $m = 1, \ldots, M$ denotes the row weight of the $m$th block row and $W_{cn}$ for $n = 1, \ldots, N$ denotes the column weight of the $n$th block column. The CWSR operation follows the RWSR operation. The RWSR operation scans $\boldsymbol{H}$ serially for the nonzero submatrices in the row-wise order, and the $m$th block row is updated serially in $W_{rm}$ clock cycles. The CWSR operation scans $\boldsymbol{H}$ serially for the nonzero submatrices in the column-wise order, and the $n$th block column is updated serially in $W_{cn}$ clock cycles. That is to say, each of the nonzero submatrices in $\boldsymbol{H}$ is processed within one clock cycle. The parameter $t$, which is given by (9), denotes the number of clock cycles taken to finish an RWSR operation or a CWSR operation. As a result, it takes $2 \times t$ clock cycles to finish a whole decoding iteration

$$t = \sum_{m=1}^{M} W_{rm} = \sum_{n=1}^{N} W_{cn}. \tag{9}$$

The input and output interfaces heavily depend on the systems, and maybe vary from one to another. In the DTMB system, the inputs are $b(=\ 127)$ 8-bit LLRs per clock cycle, and the outputs are $b(=127)$ information bits per clock cycle. As a result, it needs $L/b + (L \times \mathrm{CR})/b$ clock cycles for the input and output interfaces. Here, $L$ is the code word length, CR is the code rate, $f$ is the work frequency, and $N_{\mathrm{iter.}}$ is the maximum iteration number. Based on the assumed input and output interfaces, the decoder's throughputs $T$ can be approximately written as follows:

$$T \approx \frac{L \times f}{\frac{L}{b} + 2 \times t \times N_{\mathrm{iter.}} + \frac{(L \times CR)}{b}}. \tag{10}$$

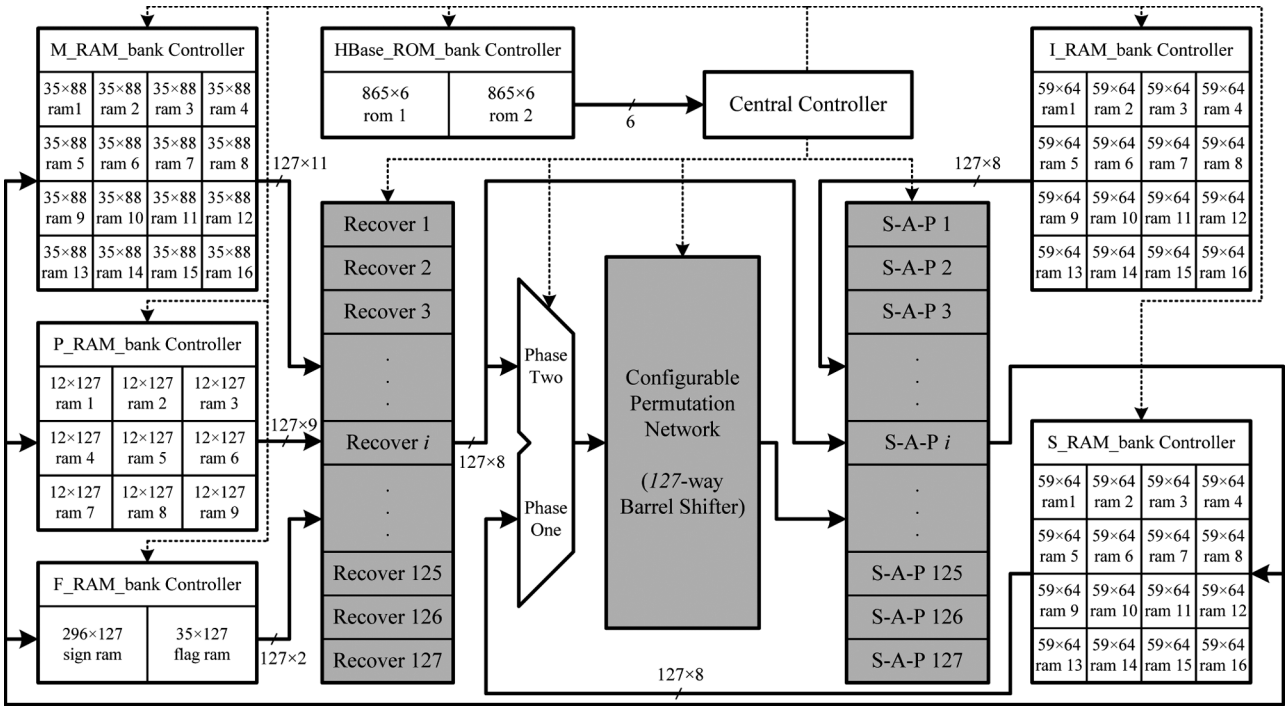



Fig. 5. Proposed TPMP decoder architecture for multirate QC-LDPC codes $(M,\ N,\ b,\ t)$.

TABLE I
THROUGHPUTS OF THE PROPOSED DECODER ARCHITECTURE FOR DIFFERENT QC-LDPC CODES $(M,\ N,\ b,\ t)$

| QC-LDPC codes in | 802.11n | 802.16e | DTMB |
|---|---|---|---|
| $f$ and $N_{iter.}$ | 125 MHz and 10 | | |
| Codeword Length $L$ | 1944 | 2304 | 7493 |
| Code Rate $CR$ | 2/3 | 3/4 | 3/5 |
| Parameter $b$ | 81 | 96 | 127 |
| Parameter $t$ | 88 | 88 | 296 |
| Estimated Throughputs $T$ | 135 Mbps | 159 Mbps | 155 Mbps |

Ignoring the clock cycles for interfaces, and fixing the work frequency and the maximum iteration number, the throughputs are directly proportional to the ratio $L/t$. Table I presents the throughputs of the proposed decoder architecture for different QC-LDPC codes $(M,\ N,\ b,\ t)$ according to (10). The proposed decoder architecture can be easily tuned up to satisfy the throughput requirements of different systems by changing the work frequency and the maximum iteration number.

### B. Proposed Iterative Decoder Architecture

The proposed iterative decoder architecture, which is shown in Fig. 5, is composed of five parts: 1) central controller; 2) memories; 3) recovers; 4) permutation network (barrel shifter); and 5) searchers, accumulators, and parity checks (S-A-Ps).

There are five banks of memories. I_RAM_bank, which is made up of 16 single-port memories with depth 59 and width 64, stores the intrinsic messages received from transferring channel. S_RAM_bank, which is made up of 16 single-port memories with depth 59 and width 64, stores the sums of all variable nodes. M_RAM_bank, which is made up of 16 single-port memories with depth 35 and width 88, stores the first minimum absolute values and the differences of all check nodes. P_RAM_bank, which is made up of nine single-port memories with depth 12 and width 127, stores the position indexes of the first minimum absolute values of all check nodes. F_RAM_bank, which is made up of a two-port sign memory with depth 296 and width 127, and a single-port flag memory with depth 35 and width 127, stores the signs of all extrinsic messages. Moreover, Hbase_ROM_bank, which is made up of two single-port memories with depth 865 and width 6, stores the position indexes (row and column indexes) and the offsets of nonzero submatrices in $\boldsymbol{H}$. When the matrix $\boldsymbol{H}$ is changed, only the Hbase_ROM_bank needs to be reconfigured.

The recovers realize the function in (8). In the previous check node updating phase, all the updated extrinsic messages are decomposed into minimum values, signs, and position indexes, which are stored into the M_RAM_bank, F_RAM_bank, and P_RAM_bank, respectively. All the extrinsic messages required in the next check node updating phase and variable node updating phase are again recovered from the components read from the corresponding memory banks.

The permutation network, which is realized by the barrel shifter, shifts the sequence of 127 messages cyclically by an offset. The RWSR operation and the CWSR operation are processed serially, so the decoder can do with only one permutation network.

Due to the serial computation and fine granularity of both scanning rounds, only one processor can deal with both of them. A total of 127 processors work in parallel and realize the functions of S-A-Ps that are shown in Fig. 5.

Table II evaluates the hardware cost of different partially parallel QC-LDPC decoder architectures applied into the DTMB system. Compared with the traditional partially parallel architectures in [20] and [21], the architecture in [22] can only reduce memory bits by about 25.5%. However, the proposed decoder architecture can largely reduce memory bits by about 59.5%. In addition, the number of memories is reduced from 355 to

TABLE II
COMPARISON WITH DIFFERENT PARTIALLY PARALLEL ARCHITECTURES FOR QC-LDPC CODES $(M, N, b, t)$ IN THE DTMB SYSTEM

| | [20], [21] | [22] | Proposed architecture |
|---|---|---|---|
| Two-Port memory bits (TP) | 308, 229 bits | 221, 869 bits | 37, 592 bits |
| Single-Port memory bits (SP) | 59, 944 bits | 59, 944 bits | 198, 653 bits |
| Total bits (2 × TP + 1 × SP) | 676, 402 bits (100%) | 503, 682 bits (74.5%) | 273, 837 bits (40.5%) |
| Memory number | 296 + 59 | 119 | 61 |
| Check-Node Function Units (CFUs) | 35 $W_r$-input CFUs | 1 $W_r$-input CFU | 127 *2*-input S-A-Ps |
| Variable-Node Function Units (VFUs) | 59 $W_c$-input VFUs | 59 *2*-input VFUs | |
| Configurability | good | good | best |
| Fitting into | regular single-rate QC-LDPC codes | | irregular multi-rate QC-LDPC codes |

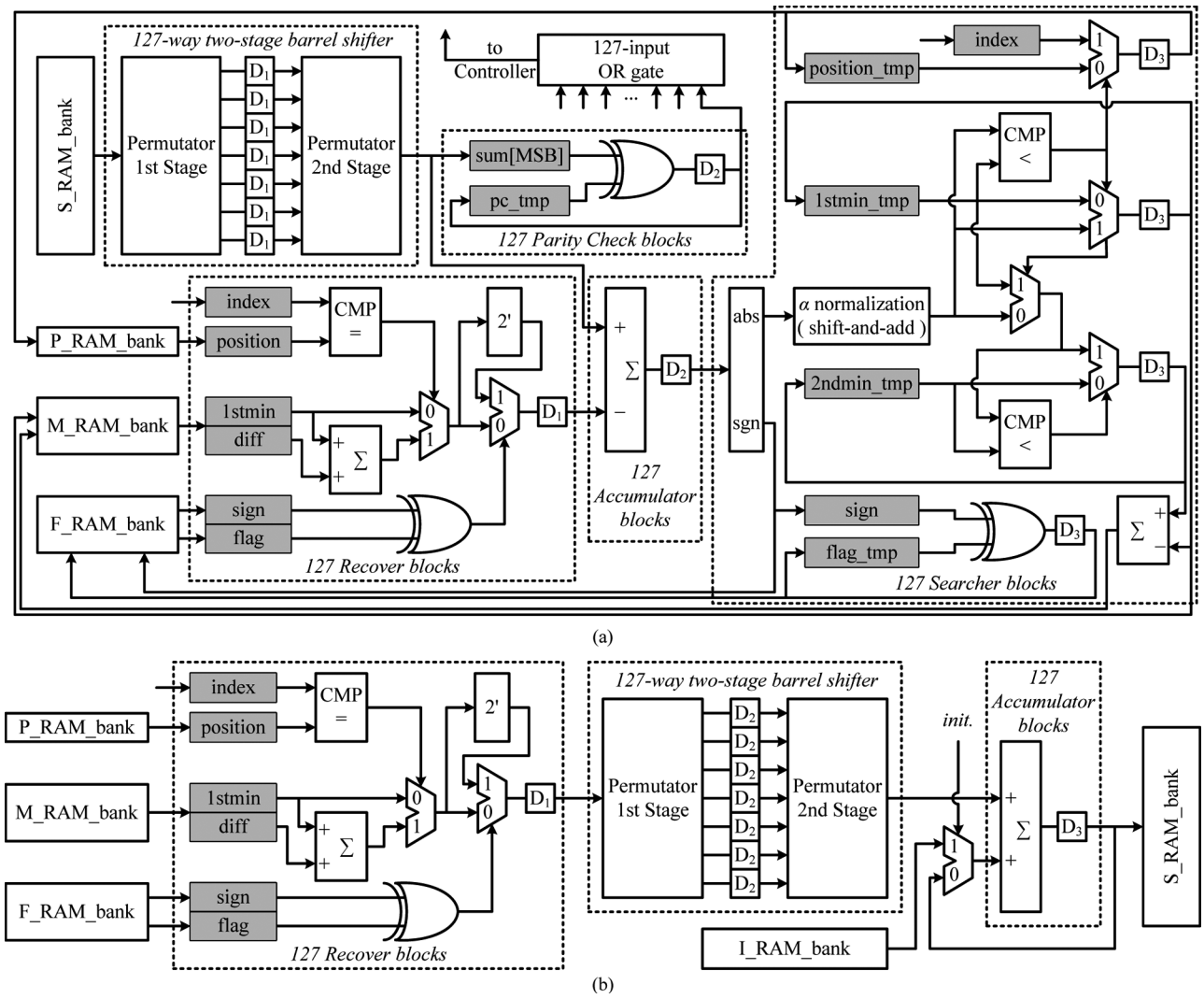



Fig. 6. Rearranged message-passing architectures: (a) RWSR architecture and (b) CWSR architecture.

61 by about 83%. It is very helpful to do back-end floor planning. Compared with the complicated check node function units and variable node function units in architectures [20]–[22], the proposed architecture only needs 127 two-input S-A-Ps in both phases. Therefore, the proposed architecture is much more efficient for area and power, has the best configurability, and fits well in irregular multirate QC-LDPC codes.

### C. RWSR and CWSR

The RWSR and the CWSR work by turns to update the extrinsic messages between check nodes and variable nodes. The architectures are shown in Fig. 6(a) and (b), respectively.

In the RWSR operation, 127 8-bit sums are first read from the S_RAM_bank, and then, get permutated through the 127-way two-stage pipeline permutation network. Meanwhile, 127 8-bit individual extrinsic messages are recovered from the minimum values, signs, and position indexes of the previous round, which are read from the M_RAM_bank, the F_RAM_bank, and the P_RAM_bank, respectively. A total of 127 8-bit updated prior messages, which are obtained from subtracting the individual extrinsic messages from the permutated sums, are decomposed into the absolute values and signs. The absolute values are normalized by the factor $\alpha$, and then, are used for updating the first and the second minimum absolute values, and the

position indexes by comparing with the temporary values in registers. The signs [most significant bits (MSBs)] are for parity check and sign updating. The intermediate updated minimum values, signs, and position indexes are temporarily latched in registers. After $W_{rm}$ for $m = 1, \ldots, M$ clock cycles, the final updated minimum values, signs, and position indexes of the $m$th block row are written back into the M_RAM_bank, the F_RAM_bank, and the P_RAM_bank, respectively. The decoder continues scanning the next block row after then.

Similarly, in the CWSR operation, 127 8-bit extrinsic messages are first recovered from the minimum values, signs, and position indexes, which are read from the M_RAM_bank, the F_RAM_bank, and the P_RAM_bank, respectively, and then, get through the 127-way two-stage pipeline permutation network. At last, the permutated extrinsic messages accumulate one by one with 127 8-bit intrinsic messages read from the I_RAM_bank. The intermediate updated sums are temporarily latched in registers. After $W_{cn}$ for $n = 1, \ldots, N$ clock cycles, the final updated sums of the $n$th block column are written back into the S_RAM_bank. The decoder continues scanning the next block column after then.

The decoder does parity check row by row in the RWSR operation, which is shown in Fig. 6. $W_{rm}$ sign bits (MSBs) pertaining to the same row feed into the logical exclusive OR gate bit by bit. After $W_{rm}$ clock cycles, all the output flag bits from different rows are carried into the logical OR gate. The whole parity-check flag bit is obtained as the RWSR operation is done. If the flag bit equals 0, the decoder converges at the right code word, and then, quits the decoding process. Meanwhile, the decoder outputs the information bits of the code word and discards the parity-check bits. All the blocks stand by until the arrival of the next data frame. Oppositely, if the flag bit equals 1, the decoder continues the next decoding iteration.

The decoder employs 4-bit unsigned digit to quantize the difference between the second minimum absolute value and the first minimum absolute value. The differences are calculated with overflow protection before they are written back into the M_RAM_bank. The second minimum absolute values are recovered again by adding the differences to the first minimum absolute values.

As shown in Fig. 6, the RWSR architecture is made up of four memory banks, 127 recovers, a 127-way two-stage pipeline permutation network, 127 accumulators, 127 searchers, and 127 parity checks, and the CWSR architecture is made up of five memory banks, 127 recovers, a 127-way two-stage pipeline permutation network, and 127 accumulators. Introducing the technique of time-division multiplexing, 127 processors operate both of them, and the hardware utility rate is improved further. Fig. 7 shows the iterative decoding mechanism for the proposed decoder. In the operation of RWSR, all the switches (SWs shown in Fig. 7) are connected with node 1. Contrarily, in the operation of CWSR, all the switches are connected with node 2, and the parity check blocks and the searcher blocks are switched off. The recover blocks, the permutation network, and the accumulator blocks are active for both scanning rounds.

As shown in Fig. 8, both scanning rounds adopt five-stage pipeline including reading from and writing into memories. The first stage $R$ is reading necessary messages from all the memory

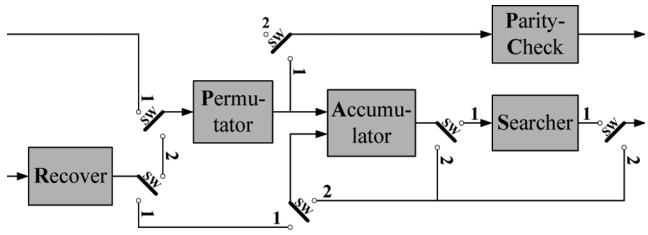



Fig. 7. Time-division multiplexing technique. Node 1 corresponds to the RWSR, node 2 corresponds to the CWSR, and SW denotes the switch.

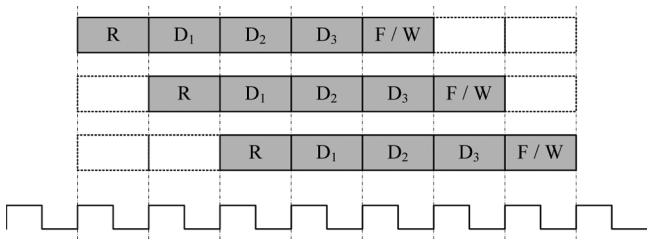



Fig. 8. Five-stage pipeline of data processing.

banks. The second, third, and fourth stages correspond to the register sets $D1$, $D2$, and $D3$, respectively, in Fig. 6. For the RWSR operation, the second stage operates the first stage of permutation network and recovers, the third stage operates the second stage of permutation network, parity checks, and accumulators, and the fourth stage operates searchers. For the CWSR operation, the second stage operates recovers, the third stage operates the first stage of permutation network, and the fourth stage operates the second stage of permutation network and accumulators. The fifth stage $F/W$ is feeding the intermediate data back or writing the final data back. During the process of scanning one block row in $\boldsymbol{H}$, the temporary data are fed back into the next logical calculation. The final updated data are written back into the corresponding memory banks as soon as it is accomplished.

### *D. Position Index and Sign Storing Optimization*

All the QC-LDPC codes $(M,\ N,\ b,\ t)$ in DTMB, 802.11n, and 802.16e feature the following two statements.

1) The row weight $W_r$ is often much smaller than the number of block columns $N$ for irregular QC-LDPC matrix $\boldsymbol{H}$ $(W_r < N)$.
2) The total number of nonzero submatrices $t$ in matrix $\boldsymbol{H}$ is almost same for different code rates $(t = \bar{W}_r \times M)$.

For 1), the position indexes of the minimum absolute value in each row can be stored with $\lceil \log_2(W_r) \rceil$-bit digit, which is much more efficient than the traditional storing scheme with $\lceil \log_2(N) \rceil$-bit digit. Therefore, the total number of memory bits for position indexes $N_{\mathrm{mbpi}}$ is calculated as in (11). Table III lists the $N_{\mathrm{mbpi}}$ for different code rates in the DTMB system by different storing schemes. Compared with the traditional storing scheme, the proposed can reduce the memory bits by 50% at most

$$N_{\mathrm{mbpi}} = M \times b \times \lceil \log_2(W_r) \rceil . \tag{11}$$

TABLE III
MEMORY BITS NEEDED FOR POSITION INDEXES BY DIFFERENT SCHEMES

| code rates | Irregular QC-LDPC codes | Row weight | Bits of row weight | Total bits of the proposed scheme | Total bits of the traditional scheme |
|---|---|---|---|---|---|
| (*N-M*)/*N* | (*M*, *N*, *b*, *t*) | $W_r$ | $\lceil \log_2(W_r) \rceil$ bits | $M \times b \times \lceil \log_2(W_r) \rceil$ bits | $M \times b \times \lceil \log_2(N) \rceil$ bits |
| 2/5 | (35, 59, 127, 275) | {7, 8} | 3 bits | 35 × 127 × 3 = 13, 335 bits (50%) | 35 × 127 × 6 = 26, 670 bits (100%) |
| 3/5 | (23, 59, 127, 296) | {12, 13} | 4 bits | 23 × 127 × 4 = 11, 684 bits (67%) | 23 × 127 × 6 = 17, 526 bits (100%) |
| 4/5 | (11, 59, 127, 294) | {26, 27} | 5 bits | 11 × 127 × 5 = 6, 985 bits (83%) | 11 × 127 × 6 = 8, 382 bits (100%) |

For 2), suppose that the parameter $t$ is constant, then the parameter $M$ would be inversely proportional to the parameter $W_r$. In other words, the decoder can maintain some kind of balance for $N_{\mathrm{mbpi}}$, no matter which code rate it is. However, if all position indexes are stored in one memory, the memory depth must be the maximum value of $M$, and the memory width must be the maximum value of $b \times \lceil \log_2(W_r) \rceil$. For the DTMB system, this storing scheme would need $35 \times 127 \times \lceil \log_2(27) \rceil = 22\,225$ bits and waste almost $22\,225 - 13\,335 = 8890$ bits. To enhance the utilization rate of memory, a novel storing scheme is presented for the multirate decoder architectures, which is shown in Fig. 9. All bits of a position index are separately stored in several memories with the same address tag. Thereby, one single-port memory with 22 225 bits is replaced by nine single-port memories with $9 \times 12 \times 127 = 13\,716$ bits. For each of them, the depth is 12 and the width is 127. The overhead of the proposed is just $13\,716 - 13\,335 = 381$ bits. As shown in Fig. 9, the nine memories are grouped into three sets, two sets, and one set for code rates 2/5, 3/5, and 4/5, respectively. For code rate 3/5, the ninth memory is no use. For code rate 4/5, the sixth, seventh, eighth, and ninth memories are no use. The central controller just selects the memory set by code rate and block row index

$$
\begin{aligned}
&\text{for } W_r = 8, \quad S'_i = \text{message}_i[\text{MSB}], \qquad i = 1, \ldots, 8 \\
&\text{begin} \\
&\quad F = S'_1, \qquad \text{clock cycle } 1 \\
&F = S'_2 \oplus F, \qquad \text{clock cycle } 2 \\
&F = S'_3 \oplus F, \qquad \text{clock cycle } 3 \\
&F = S'_4 \oplus F, \qquad \text{clock cycle } 4 \\
&F = S'_5 \oplus F, \qquad \text{clock cycle } 5 \\
&F = S'_6 \oplus F, \qquad \text{clock cycle } 6 \\
&F = S'_7 \oplus F, \qquad \text{clock cycle } 7 \\
&F = S'_8 \oplus F, \qquad \text{clock cycle } 8 \\
&S_i = F \oplus S'_i, \qquad \text{for } i = 1, \ldots, 8 \\
&\text{end.}
\end{aligned} \tag{12}
$$

The signs of the extrinsic messages are updated according to (12) in the proposed decoder architecture. The traditional way of sign storing in Fig. 10(a) is to send the final updated signs $S_i$ into the sign memory. For the parameters $M$ and $W_r$ vary from one code rate to another, this storing scheme needs large numbers of memory bits and registers, which is shown in Table IV, where TP denotes the two port and SP denotes the single port.

An optimized storing scheme, which is much more befitting for the proposed multirate decoder architecture, is presented in Fig. 10(b). In the proposed architecture, the message updating of each block row is processed by serial scanning in $W_r$ clock cycles. However, all the valid signs $S_i$ are just known after $W_r$ clock cycles. To avoid the usage of many registers, the decoder

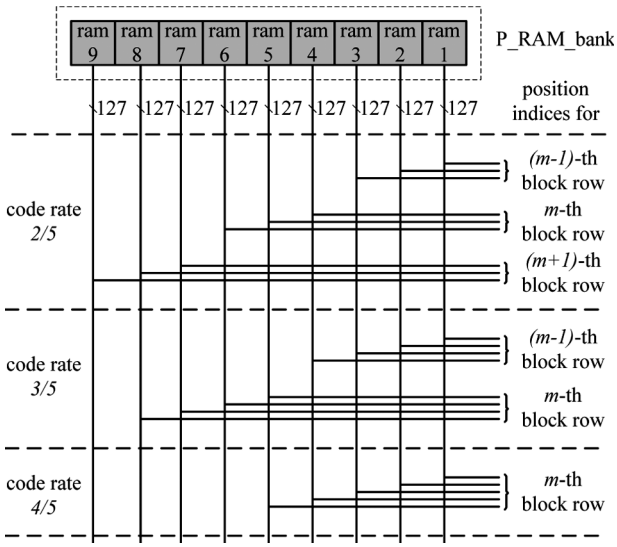



Fig. 9. Position index storing optimization for multirate QC-LDPC codes.

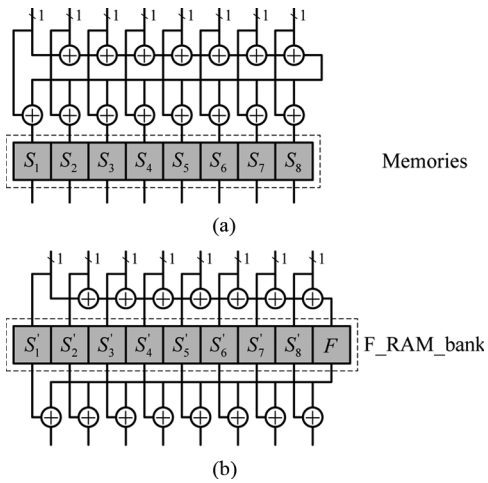



Fig. 10. Sign storing optimization: (a) traditional scheme and (b) proposed scheme. Here, $W_r$ equals 8.

TABLE IV
DIFFERENT SIGN STORING SCHEMES FOR MULTIRATE QC-LDPC CODES IN THE DTMB SYSTEM

| | Memory bits (2 × TP + 1 × SP) | Registers |
|---|---|---|
| Traditional | 1 × max(*M*) × *b* × max($W_r$) = 120, 015 bits | *b* × max($W_r$) = 3, 429 |
| Proposed | 2 × max(*t*) × *b* + 1 × max(*M*) × *b* = 79, 629 bits | *b* × 1 = 127 |
| Reduced by | ≈ 34% | ≈ 96% |

stores the intermediate signs $S'_i$ and $F$ instead of the valid signs $S_i$. Although it requires additional $\max(M) \times b = 4445$ bits for $F$, the proposed scheme can deal with multirate architectures

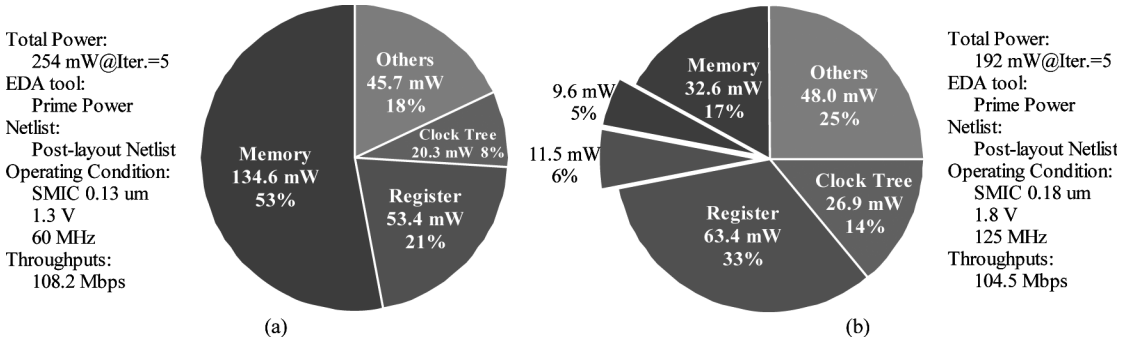



Fig. 11. Power distribution of different architectures: (a) traditional partially parallel architectures [21] and (b) proposed architecture.

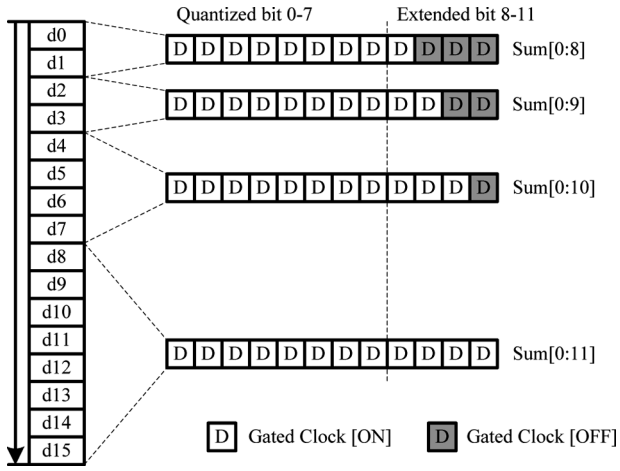



Fig. 12. Dynamic power controlling technique of clock gating.

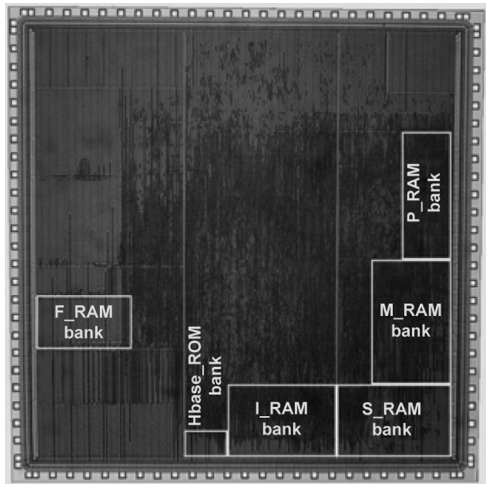



Fig. 13. Die photograph of the FEC decoder in the DTMB system.

very well, and in total, it can save memory bits by about 34% and registers by about 96%, which are shown in Table IV.

### *E. Power Analysis and Clock Gating*

A traditional partially parallel decoder architecture [21] for the DTMB system is implemented in the SMIC 0.13 $\mu$m prior to the proposed one. This decoder architecture works at 60 MHz and 1.3 V, and attains a throughput of 108.2 Mb/s at 15 decoding iterations. To analyze the power distribution of the different architectures more accurately, the postlayout netlists with the parasitic annotation are used by the synopsys electronic design automation (EDA) tool, the Prime Power. Fig. 11(a) shows the power distribution of the traditional partially parallel architecture [21]. It has revealed that about 74% of the total chip power is consumed by continuous memory accesses and flip-flop overturns. Therefore, decreasing the memory size and register number is the most efficient way to lower the decoder's power dissipation. The power distribution of the proposed partially parallel architecture is shown in Fig. 11(b). For much less number of memories and much less memory accesses per iteration, the proportion of power dissipation for all memory blocks is reduced from 53% to $22\%(= 17\% + 5\%)$. However, the proposed architecture also uses large numbers of registers to latch temporary results, and the proportion for this part goes up from 21% to $39\%(= 33\% + 6\%)$.

When the estimated code word $\hat{\boldsymbol{c}}$ satisfies the parity-check matrix $\boldsymbol{H}$, the decoder terminates the iterative decoding process, and gates off all the clocks of memory banks and ALUs. They would be again activated by the central controller as soon as the next data frame arrives.

For the multirate decoder architecture, the hardware utility rate must be less than 1, and some parts of the decoder may not operate in the course of iterative decoding. Thereby, the clock gating is introduced for the proposed architecture to lower the power dissipation of unnecessary memory accesses and register overturns. For the serial updating, and the asymmetry of RWSR operation and CWSR operation, not all the memories and registers are always in active state. The clocks of the memories and registers in the idle state would be gated off.

About 39% of the total power is consumed by the large numbers of registers in the proposed decoder. Fig. 12 shows an example of clock gating to control power dissipation of some registers dynamically. The maximum column weight $W_c$ in $\boldsymbol{H}$ is 16. In the variable node updating phase, it employs 12-bit quantization with 4-bit extension for the sum of each column to guarantee high-precision requirements of the proposed decoder. Therefore, it needs 12 registers for 12-bit sum. However, the weights are only 3 for most columns, which is much smaller than 16. The unnecessary registers are gated off dynamically in the serial accumulation process. In the check node updating phase, it only uses eight registers for 8-bit extrinsic message, and the other four registers are gated off all along.

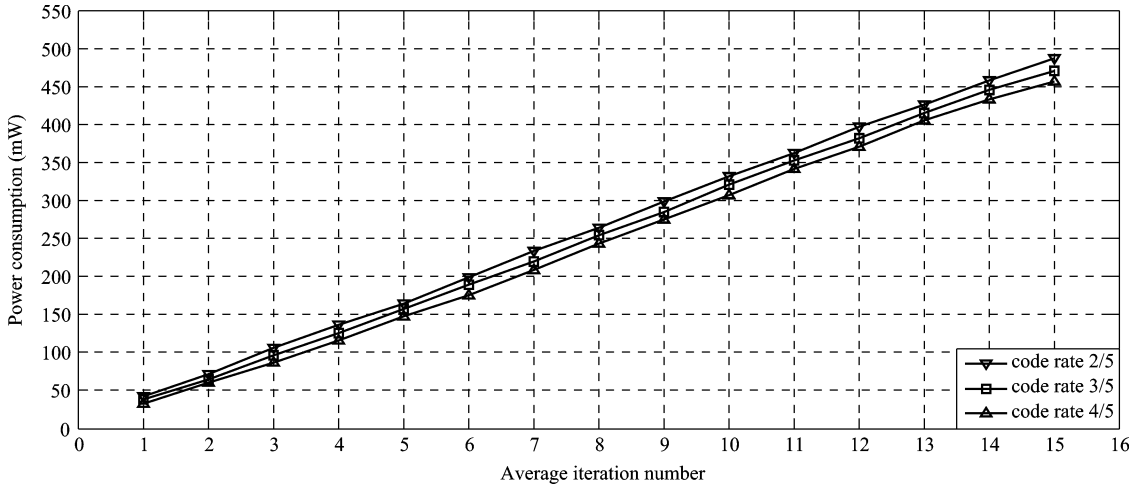



Fig. 14. Power dissipation versus average iteration number for different code rates.

Fig. 11(b) shows that the clock gating can save 5% of the total power dissipation for memory accesses (the upper part excluded from pie diagram), and 6% for register overturns (the lower part excluded from pie diagram).

## IV. Implementation Results

The decoder is fabricated in the SMIC 0.18-$\mu$m six-metal-layer standard CMOS technology. Fig. 13 shows the die photograph of the FEC decoder in the DTMB system, including the proposed QC-LDPC decoder, deinterleaver, demapper, Bose–Chaudhuri–Hocquenghem (BCH) decoder, and synchronized parallel interface (SPI). Due to the large numbers of memory accesses in both phases, all memories are placed around the chip to balance the power distribution.

Fig. 14 presents power dissipation of the proposed decoder versus average iteration number. Due to more memory accesses to the M_RAM_bank, the P_RAM_bank, and the F_RAM_bank for lower code rates, the power dissipation is a little more than that for higher code rates. When it works at high SNR, the power dissipation is less than 100 mW.

Table V lists VLSI implementation results of the proposed decoder and the previous decoder with traditional partially parallel architecture [21] for QC-LDPC codes in the DTMB system. The chip in this study can run at a peak frequency of 250 MHz with a throughput of about 209 Mb/s at 15 iterations. The memory bits of the proposed decoder are about 236 000 bits at 8-bit quantization. In the DTMB system, the decoder operates at 125 MHz, and dissipates an average power of 486 mW at 15 iterations. There are about 580 000 logic equivalent gates, which occupy an area of about 9.76 mm$^2$. The work frequency and the maximum iteration number can be adjusted to satisfy the throughput requirements of different systems.

The area of the two-port memory is about 1.5 times larger than the single port with the same memory size. Table V indicates that most of them in the proposed architecture are the single port. The size ratio between the single-port memories and the two-port memories is 5.28/1, which is much greater than 1/5.14 for the traditional architecture [21]. That is to say, the area of the memories becomes smaller as the ratio becomes larger.

TABLE V
Comparison of Implementation Results Between Proposed Architecture and Previous Architecture [21] in the DTMB System

| Parameters | In this work | In previous work [21] |
|---|---|---|
| Structure | Partially Parallel | Partially Parallel |
| Parallel Degree | 127 | 35, 59 |
| Code Rates | 2/5, 3/5, 4/5 | 2/5, 3/5, 4/5 |
| Work Frequency | 125 MHz | 60 MHz |
| Maximum Iteration Number | 15 | 15 |
| Throughputs | 104.5 Mbps | 108.2 Mbps |
| Quantization Bits | 8 | 8 |
| Memory Bits | 236, 245 bits | 368, 173 bits |
| Single-Port/Two-Port | 5.28/1 | 1/5.14 |
| Number of Memories | 61 | 296 + 59 |
| Memory Access Efficiency (*MAE*) | 60.0 bits/Bit/Iter. | 188.6 bits/Bit/Iter. |
| Combinational Logic Equivalent Gates | 100 K | 400 K |
| Non-Combinational Logic Equivalent Gates | 480 K | 2, 100 K |
| Number of Registers | 8, 237 | 10, 781 |
| Permutation Network | Necessary | Unnecessary |
| Reconfigurable | Yes | No |
| Routing Complexity | Low | High |
| Core Area | 9.76 mm$^2$ | 14.4 mm$^2$ |
| Power Dissipation | 164 mW @ Iter .= 5 | 200 mW @ Iter. = 5 |
| Technology | SMIC 0.18 um, 1.8 V | SMIC 0.13 um, 1.3 V |

The *memory access efficiency (MAE)* in Table V, which is defined in (13), is the average number of memory accesses per variable node per iteration, where $N_{\rm ma}$ is the total number of memory accesses per iteration and $L$ is the code word length

$$\text{MAE} = \frac{N_{\rm ma}}{L}. \tag{13}$$

Compared with the architecture in [21], the memory bits are reduced by 40%. The MAE decreases from 188.6 bits to 60 bits by about 68%. Total number of logic equivalent gates is only about 580 000 reduced by about three fourths. Additionally the architecture has much better flexibility for different code rates and code word lengths, lower routing complexity, smaller area, and lower power consumption.

TABLE VI
COMPARISON WITH LDPC DECODERS IN OTHER SYSTEMS

| Parameters | In this work | [25] | [26] | [11] | [27] |
|---|---|---|---|---|---|
| Structure | Partially Parallel | Partially Parallel | Partially Parallel | Partially Parallel | Fully Parallel |
| Codeword Lengths | 7493 | 576 - 2304 | 576 - 2304 | 2048 | 1024 |
| Code Rates | 2/5, 3/5, 4/5 | 1/2, 2/3A, 2/3B, 3/4A, 3/4B, 5/6 | 1/2 | 8/16:1/16:14/16 | 1/2 |
| Number of Nonzero Sub-Matrices | 275, 296, 294 | 76, 80, 81, 85, 88 | 76 | 96 | Random |
| Size of Sub-Matrices | 127 × 127 | 24 × 24 - 96 × 96 | 24 × 24 - 96 × 96 | 64 × 64 | - |
| Distribution Density of '1' in $\boldsymbol{H}$ | 0.00356 | 0.0087 | 0.00274 | 0.00293 | - |
| Average Column Weight | 5.0 | 3.7 | 3.2 | 3 | - |
| Work Frequency | 125 MHz | 150 MHz | 83.3 MHz | 125 MHz | 64 MHz |
| Maximum Iteration Number | 15 | 20 | 2 - 8 | 10 | 64 |
| Throughputs | 104.5 Mbps | 105 Mbps | 60 - 222 Mbps | 640 Mbps | 1 Gbps |
| Quantization Bits | 8 | 6 | 8 | 4 | 4 |
| Memory Bits | 236, 245 bits | 75, 856 bits | 76, 800 bits | 51, 680 bits | 34, 816 bits |
| Single-Port/Two-Port | 5.28/1 | - | 1/3.17 | 1/1.06 | - |
| Memory Usage Efficiency (*MUE*) | 0.65 | 1.18 | 1 | 1.6 | - |
| Power Dissipation | 486 mW | 264 mW | 52 mW | 787 mW | 690 mW |
| Power Efficiency (*PE*) | 310 pJ/Bit/Iter. | 125 pJ/Bit/Iter. | 108 pJ/Bit/Iter. | 123 pJ/Bit/Iter. | 10.9 pJ/Bit/Iter. |
| Estimated *PE* in 0.18 um | 310 pJ/Bit/Iter. | 1000 pJ/Bit/Iter. | 305 pJ/Bit/Iter. | 492 pJ/Bit/Iter. | 15.5 pJ/Bit/Iter. |
| Normalized *PE* (*NPE*) in 0.18 um | 62 pJ/Bit/Iter. | 270 pJ/Bit/Iter. | 95 pJ/Bit/Iter. | 164 pJ/Bit/Iter. | - |
| Logic Equivalent Gates | 580 K | 970 K | 420 K | 220 K | 1, 750 K |
| Area | 9.76 $mm^2$ (core area) | 6.25 $mm^2$ | 8.29 $mm^2$ | 14.3 $mm^2$ | 52.5 $mm^2$ |
| Reconfigurable | Yes | Yes | Yes | Yes | No |
| Technology | 0.18 um, 1.8 V | 90 nm, 1.0 V | 0.13 um, 1.2 V | 0.18 um, 1.8 V | 0.16 um, 1.5 V |

Table VI lists comparison results among different LDPC decoders of different systems. Compared with the other four LDPC codes, the codes in the DTMB system are much more complex. First, the code word lengths are much longer. Second, the number of ''1'' in $\boldsymbol{H}$ is $296 \times 127$ at most, which is about 4.4 times of the number in [25] and [26], and about six times of the number in [11].

Here, two normalized parameters to compare different LDPC decoders are given. The first parameter is the *memory usage efficiency (MUE)*, which is defined in (14), where $N_{\mathrm{mb}}$ denotes the total number of memory bits and $N_{\mathrm{quant.}}$ denotes the number of quantization bits

$$\mathrm{MUE} = \frac{N_{\mathrm{mb}}}{(L + b \times t) \times N_{\mathrm{quant.}}}. \tag{14}$$

In (14), the denominator is the total number of memory bits in traditional partially parallel architectures using the original logarithmic domain TPMP algorithm. When all the memory resources are normalized to this level, the proposed decoder's MUE is only 0.65, which is much smaller than the numbers in [11], [25], and [26]. Furthermore, the majority of memories are the single port. The size ratio between the single-port memories and the two-port memories is 5.28/1 in this paper. However, it is about 1/3.17 in [26] and 1/1.06 in [11]. In other words, the chip area occupied by the memory bits is decreased furthest for the proposed decoder architecture. The memory usage is much more efficient than the other three decoders.

The second parameter is the *power efficiency (PE)*, which is defined in (15). $P$ is the total power dissipation, $T$ is the throughputs, and $N_{\mathrm{iter.}}$ is the maximum iteration number

$$\mathrm{PE} = \frac{P}{T \times N_{\mathrm{iter.}}}. \tag{15}$$

The proposed decoder consumes an average power of 310 pJ per variable node per iteration at 1.8 V. Table VI lists the reference values of PEs for different LDPC decoders. Since the technologies are not uniform, it is meaningless to compare them with one another directly.

The total power consumption $P$ of the circuits is the sum of the dynamic power $P_{\mathrm{dyn}}$ and the static power $P_{\mathrm{sta}}$, as shown in (16). The static power, which is always much smaller than the dynamic power, can be ignored. The dynamic power is made up of the switch power $P_{\mathrm{sw}}$ and the short-circuit power $P_{\mathrm{sc}}$. The variables $C_L$ and $C_{sc}$ denote the total load capacitance and the total equivalent short-circuit capacitance in the circuits. The variable $V_{\mathrm{DD}}$ is the work voltage and the variable $f_{0\to1}$ is the frequency of switching activities from 0 to 1 for logic gates.

$$\begin{aligned} P &= P_{\mathrm{dyn}} + P_{\mathrm{sta}} \approx P_{\mathrm{dyn}} = P_{\mathrm{sw}} + P_{\mathrm{sc}} \\ &= V_{\mathrm{DD}}^2 \times C_L \times f_{0\to1} + V_{\mathrm{DD}}^2 \times C_{\mathrm{sc}} \times f_{0\to1} \\ &= V_{\mathrm{DD}}^2 \times (C_L + C_{\mathrm{sc}}) \times f_{0\to1}. \end{aligned} \tag{16}$$

According to the constant scaling rule, the work voltage $V_{\mathrm{DD}}$, the capacitances $C_L$ and $C_{\mathrm{sc}}$, and the circuit delay $t_d$ are scaled down as in (17) in the successive process generation. Here, $\lambda$ is the constant scaling factor whose typical value is $1/\sqrt{2}$. As shown in (17), the decrease of circuit delay $t_d$ would result in increase of the maximum work frequency $f_{\mathrm{max}}$. Here, we do not consider the relationship between the circuit delay and the maximum work frequency, and suppose that all the throughputs in decoders [11], [25]–[27] could be attained with the same work frequencies in the 0.18-$\mu$m technology. Once the frequency of switching activities $f_{0\to1}$ for all logical gates is constant, the total power consumption $P'$ is scaled down by the cubic factor $\lambda^3$, as shown in (18)

$$\begin{aligned} V'_{\mathrm{DD}} &= \lambda \times V_{\mathrm{DD}} \\ C'_L &= \lambda \times C_L \qquad C'_{\mathrm{sc}} = \lambda \times C_{\mathrm{sc}} \end{aligned} \tag{17}$$

$$t'_d = \lambda \times t_d \qquad f'_{\max} = \frac{1}{t'_d} = \frac{f_{\max}}{\lambda}$$
$$P' \approx (\lambda \times V_{\mathrm{DD}})^2 \times (\lambda \times C_L + \lambda \times C_{\mathrm{sc}}) \times f_{0\to 1} = \lambda^3 \times P. \tag{18}$$

Therefore, all the PEs can be estimated in the $0.18$-$\mu$m technology based on the constant scaling rule. For example, if both of the work voltage and the total capacitance are scaled up from $0.13$-$\mu$m technology to $0.18$-$\mu$m technology by the factor $\sqrt{2}$, then the total power consumption is scaled up approximately by the factor $\sqrt{8}$. The reference values of PEs in the $0.18$-$\mu$m technology for different LDPC decoders are listed in Table VI.

If considering the density of ''1'' and the average column weight $\overline{W_c}$ in $\boldsymbol{H}$, the *normalized power efficiency (NPE)* is defined by (19). The symbol denotes the estimated power consumption in the $0.18$-$\mu$m technology. Actually, NPE is the average power consumption per ''1'' in $\boldsymbol{H}$ per iteration. The decoder in this paper has the smallest NPE than the other three partially parallel decoders [11], [25], [26]

$$\mathrm{NPE} = \frac{[P]_{0.18\ \mu\mathrm{m}}}{T \times N_{\mathrm{iter.}} \times \overline{W_c}}. \tag{19}$$

## V. Conclusion

A configurable, area-efficient, and low-power decoder for 7493-bit irregular multirate QC-LDPC codes has been presented in this paper. It dissipates an average power of 486 mW, and attains a throughput of 104.5 Mb/s when operated at 125 MHz and 1.8 V.

Based on the features of the second minimum absolute values, the decoder adopts 4-bit unsigned digit to quantize the difference between the second minimum absolute value and the first minimum absolute value, which leads to 5.64% decrease of the memory bits.

The decoder absorbs fully the advantage of fine granularity of the proposed architecture, and performs both the RWSR operation and the CWSR operation with the same hardware processor. Additionally, much more efficient position index and sign storing schemes are introduced in the decoder.

To reduce the total power dissipation further, the technique of clock gating is adopted in the decoder, which can reduce the power dissipation of memories and registers by about 11%. Especially, in the phase of CWSR, the technique of dynamic power controlling is very flexible and efficient to reduce the registers' power consumption.

The decoder is configurable and can be easily tuned up to different code rates and code word lengths. It can also have applications in other high-speed communication systems very flexibly by changing the work frequency and the maximum iteration number.

## References


[1] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.

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

[3] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[4] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*. Cambridge, MA: MIT Press, 1972.

[5] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[6] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Dept. Elect. Eng., Univ. Linkoping, Linkoping, Sweden, 1996.

[7] X.-Y Hu, E. Eleftherious, D.-M Arnold, and A. Dholakia, "Efficient implementation of the sum-product algorithm for decoding LDPC codes," in *Proc. IEEE GLOBECOM*, Nov. 2001, vol. 2, pp. 1036–1036E.

[8] J. Chen and M. C. Fossorier, "Decoding low-density parity-check codes with normalized APP-based algorithm," in *Proc. IEEE GLOBECOM*, Nov. 2001, vol. 2, pp. 1026–1030.

[9] J. Chen, "Reduced complexity decoding algorithms for low-density parity-check codes and turbo codes," Ph.D. dissertation, Dept. Elect. Eng., Univ. Hawaii, Honolulu, 2003.

[10] M. M. Mansour and N. R. Shanbhag, "Turbo decoder architectures for low-density parity-check codes," in *Proc. IEEE GLOBECOM*, Nov. 2002, pp. 1383–1388.

[11] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 634–698, Mar. 2006.

[12] E. Yeo, P. Pakzad, B. Nikoli, and V. Anantharam, "High throughput low-density parity-check decoder architectures," in *Proc. IEEE GLOBECOM*, Nov. 2001, vol. 5, pp. 3019–3024.

[13] H. Zhong, W. Xu, N. Xie, and T. Zhang, "Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording," *IEEE Trans. Magn.*, vol. 43, no. 12, pp. 4117–4122, Dec. 2007.

[14] Y. Chen and K. K. Parhi, "Overlapped message passing for quasi-cyclic low density parity check codes," *IEEE Trans. Circuits Syst.*, vol. 51, no. 6, pp. 1106–1113, Jun. 2004.

[15] N. Chen, Y. Dai, and Z. Yan, "Partly parallel overlapped sum-product decoder architectures for quasi-cyclic LDPC codes," in *Proc IEEE SIPS*, Oct. 2006, pp. 220–225.

[16] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[17] F. Guilloud, "Generic architectures for LDPC codes decoding," Ph.D. dissertation, Dept. Elect. Eng. Telecomm., Telecom Paris, Paris, France, 2004.

[18] Mansour and M. M.Shanbhag, "N. R. low-power VLSI decoder architectures for LDPC codes," in *Proc. ISLPED*, 2002, pp. 284–289.

[19] T. Zhang and K. K. Parhi, "A 54 Mbps (3, 6)-regular FPGA LDPC decoder," in *Proc IEEE SIPS*, Oct. 2002, pp. 127–132.

[20] M. Karkooti and J. R. Cavallaro, "Semi-parallel reconfigurable architectures for real-time LDPC decoding," in *Proc. Int. Conf. ITCC*, 2004, vol. 1, pp. 579–585.

[21] C. Yun, Z. Xiao-yang, L. Yi-Fan, X. Bo, and D. Yun-Song, "VLSI design of an irregular LDPC decoder in DTMB," *J. Commun.*, vol. 28, no. 8, pp. 61–66, Aug. 2007.

[22] Z. Wang and Z. Cui, "A memory efficient partially parallel decoder architecture for QC-LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 4, pp. 483–488, Apr. 2007.

[23] Y. Chen and D. Hocevar, "A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder," in *Proc. IEEE GLOBECOM*, Dec. 2003, vol. 1, pp. 113–117.

[24] D. MacKay and M. Postol, "Weaknesses of margulis and ramanujan-margulis low-density parity-check codes," *Electron. Notes Theor. Comput. Sci.*, vol. 74, pp. 97–104, 2003.

[25] C.-H. Liu, S.-W. Yen, C.-L. Chen, H.-C. Chang, C.-Y. Lee, Y.-S. Hsu, and S.-J. Jou, "An LDPC decoder chip based on self-routing network for IEEE 802.16e applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 684–694, Mar. 2008.

[26] X.-Y. Shih, C.-Z. Zhan, C.-H. Lin, and A.-Y. Andy Wu, "An $8.29$ mm$^2$ 52 mW multi-mode LDPC decoder design for mobile wiMAX system in 0.13 CMOS process," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 672–683, Mar. 2008.

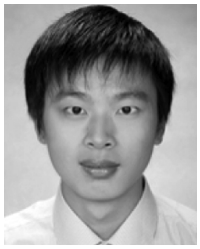
[27] A. J. Blanksby and C. J. Howland, "A690-mW1-Gb/s1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002.

**Bo Xiang** received the B.S. degree in microelectronics from Sichuan University, Chengdu, China, in 2005. He is currently working toward the Ph.D. degree in microelectronics in the State Key Laboratory of Application-Specific IC (ASIC) and System, Department of Microelectronics, Fudan University, Shanghai, China.

His research interests include wireless communication systems and their VLSI architecture design, in particular, the channel coding and decoding algorithms, and their VLSI implementations.

**Rui Shen** received the B.S. degree in electrical engineering from Tongji University, Shanghai, China, in 2006. She is currently working toward the M.S. degree in microelectronics in the State Key Laboratory of Application-Specific IC (ASIC) and System, Department of Microelectronics, Fudan University, Shanghai.

Her research interests include wireless communication systems and their VLSI architecture design.



**An Pan** received the B.S. degree from East China Normal University, Shanghai, China, in 2006. He is currently working toward the Masters' degree in microelectronics in the State Key Laboratory of Application-Specific IC (ASIC) and System, Department of Microelectronics, Fudan University, Shanghai, China.

His research interests include digital signal processing, orthogonal frequency-division multiplexing systems, and wireless transmission communications, in particular, the channel estimation and equalization for the high-definition TV systems.

**Dan Bao** received the B.S. degree in electronics from Beijing University of Aeronautics and Astronautics, Beijing, China, in 2005. He is currently working toward the Ph.D. degree in microelectronics in the State Key Laboratory of Application-Specific IC (ASIC) and System, Department of Microelectronics, Fudan University, Shanghai, China.

His research interests include ASIC designs, and interactions between algorithms and VLSI architectures in broadband wireless transmission systems.

**Xiaoyang Zeng** (M'06) received the B.S. degree from Xiangtan University, Xiangtan, China, in 1992, and the Ph.D. degree from Changchun Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, Beijing, China, in 2001.

Since 2007, he has been a Full Professor and the Director of the State Key Laboratory of Application-Specific IC (ASIC) and System, Fudan University, Shanghai, China, where he was a Postdoctoral Researcher from 2001 to 2003, and later an Associate Professor in the Department of Microelectronics. His research interests include information security chip design, VLSI signal processing, and communication systems.

Prof. Zeng is the Steering Committee Member of the Asia and South Pacific Design Automation Conference, and the TPC member of the IEEE Asian Solid-State Circuits Conference and the International Conference on ASIC (ASICON).