A Flexible LDPC Decoder Architecture Supporting TPMP and TDMP Decoding Algorithms**

Shuangqu HUANG[†], Xiaoyang ZENG[†], *Nonmembers*, and Yun CHEN^{†*a)}, *Member*

SUMMARY In this paper a programmable and area-efficient decoder architecture supporting two decoding algorithms for Block-LDPC codes is presented. The novel decoder can be configured to decode in either TPMP or TDMP decoding mode according to different Block-LDPC codes, essentially combining the advantages of two decoding algorithms. With a regular and scalable data-path, a Reconfigurable Serial Processing Engine (RSPE) is proposed to achieve area efficiency. To verify our proposed architecture, a flexible LDPC decoder fully compliant to IEEE 802.16e applications is implemented on a 130 nm 1P8M CMOS technology with a total area of 6.3 mm² and maximum operating frequency of 250 MHz. The chip dissipates 592 mW when operates at 250 MHz frequency and 1.2 V supply.

key words: iterative decoding, LDPC codes, reconfigurable architecture, TDMP, TPMP

1. Introduction

Since the Low-Density Parity-Check (LDPC) [1] codes have near-capacity decoding performance and very high decoding throughput, they have been employed as Forward Error Correction (FEC) coding scheme in many transmission standards for wireless communication such as wireless LAN (IEEE 802.11n) [2], WiMAX (IEEE 802.16e) [3], DVB-S2 [4], DTMB [35] and CMMB [5]. LDPC codes are also considered for error correction coding in virtually all next generation communication systems. The decoding schedule of LDPC codes can be carried out following a concept of iterative decoding, which well suited for parallel hardware implementation. The most typical decoding algorithm is Two-Phase Message Passing (TPMP) algorithm [1], which involves check-node update and variable-node update as two sequential phase schedule. In addition to TPMP algorithm, authors in [7] introduced a concept of Turbo-Decoding Message Passing (TDMP, also called as layered belief propagation) algorithm, which allows updating both check nodes and variable nodes concurrently. As we known, TDMP algorithm achieves about two times faster decoding convergence and significant memory advantage when compared to TPMP algorithm. However, this novel decoding algorithm is not suitable for unstructured Block-LDPC codes [8] unless a certain amount of performance and throughput are lost.

Almost all of the research on LDPC decoder design so far has focused on high throughput, low area and low power consumption, supporting multiple code rates and code lengths, in which specific optimizations are made to improve the decoder performance [9]–[14], [22], [24]–[31]. On the other hand, techniques such as ASIP, NOC have been adopted to implement generic and reconfigurable LDPC decoder [15], [32]–[34]. While fully parallel LDPC decoder design [9] suffers from complex interconnect issues, various semi-parallel implementations based on structured Block-LDPC codes have been proposed [10]–[13], which can achieve a good tradeoff between hardware complexity and decoding throughput. Decoders in [10] and [11] take advantage of inherent parallelism in TPMP algorithm and overlapped structure in the LDPC codes and can support multi-mode. Decoders in [12] and [13] take advantage of the property of high convergence rate in TDMP algorithm and achieve high throughput. However, almost all of the LDPC decoders adopting TDMP algorithm will suffer from data-collision when decode a class of unstructured Block-LDPC codes such as HS-LDPC codes [5], EG-LDPC codes [14] and IRA-LDPC codes [4]. To solve this problem, [24] proposes a method by reordering the matrix to reduce the number of data-collision, at cost of performance and throughput. A serial decoder based on TPMP algorithm enables decoding of arbitrarily designed LDPC codes [15]. However, this decoder has not been chip implemented and the throughput is less than architectures optimized for specific codes. However, there are no good solutions to make good trade-off between high throughput (especially high convergence in TDMP algorithm) and supporting any existing Block-LDPC codes. On the other hand, all of the existing LDPC decoders only adopt one decoding algorithm within TPMP, TDMP or their modifications, which can't make good use of decoding flexibility in TPMP algorithm and high convergence rate in TDMP algorithm.

Nowadays, as multi-mode/standard LDPC decoder within one hardware platform has become more and more important, we aim to design a flexible architecture which can support multiple code rates and code lengths and can decode any existing Block-LDPC codes, especially suitable for multi-standard schemes. Meanwhile, in order to make a good trade-off between high throughput and supporting all kinds of Block-LDPC codes, which mainly means making best use of high convergence rate in TDMP algorithm and achieving decoding flexibility in TPMP to decode all existing unstructured Block-LDPC codes, the proposed architec-

Manuscript received April 25, 2011.

Manuscript revised August 15, 2011.

[†]The authors are with State Key Laboratory of ASIC & System, Fudan University, Shanghai, P.R. China, 201203.

*Corresponding author.

**This work is supported by National Science and Technology Major Project of China under Grant No.2011ZX03003-003-03.

a) E-mail: chenyun@fudan.edu.cn

DOI: 10.1587/transinf.E95.D.403

ture can decode in either TPMP or TDMP decoding mode according to the kind of Block-LDPC codes. For example, a multi-standard LDPC decoder supporting IEEE 802.16e standard has been fabricated in this paper, which can operate in TDMP mode to maximize throughput when decoding QC-LDPC codes and can operate in TPMP mode to avoid data-collision when decoding HS-LDPC codes like in CMMB standard. As the high area/power efficiency design is critical for LDPC decoder, we deploy the following techniques to implement the proposed architecture:

1) The schedule of TPMP algorithm is rearranged to be easy to share hardware resources with TDMP algorithm, including memory blocks, basic processing units and so on. Moreover, The Modified Min-Sum (MSA) [16] algorithm is adopted in check-node operation to reduce decoding complexity. A method of sub-matrix dispatch is proposed to decode unstructured Block-LDPC codes.

2) A RSPE which has scalable data-path and can be programmed to perform computations in two decoding algorithms is proposed. Meanwhile, the RSPE can decode in different mode with 6-stage pipeline to achieve a maximum throughput. Furthermore, The basic processing units in RSPE are working by time-division multiplexing to gain hardware utilization efficiency (HUE) and the unused processing units can be deactivated to reduce overall power consumption.

3) To further improve area/power efficiency, a) only single-port memory is required by exclusive write/read operations and register-buffering, b) a memory-sharing scheme has been deployed in two decoding algorithms, c) extrinsic messages in two algorithms are stored in a compressed form [17], d) memory access rate can be reduced by data buffering.

The remainder of this paper is organized as follows. Section 2 first gives a brief overview of Block-LDPC codes, and then comparison and combination between RS-TPMP and TDMP have been discussed. Section 3 proposes a flexible LDPC decoder hardware architecture and mainly addresses its primary processing unit RSPE. Section 4 shows the chip implementation results of the proposed LDPC decoder. Finally, some concluding remarks are given in Sect. 5.

2. LDPC Codes and Decoding Algorithm

2.1 LDPC Codes

Low-Density Parity-Check codes are a class of linear block code defined by an $M \times N$ parity check matrix, where M represents the number of LDPC check bits and N represents the number of LDPC code-word bits. The parity check matrix can be represented by a bipartite graph called Tanner graph [18]. There are two types of nodes in the Tanner graph, called check nodes and variable nodes. Each of the check nodes c_m ($m = 1, \dots, M$) is connected to variable node v_n ($n = 1, \dots, N$), where elements of each row in the parity check matrix are one. Figure 1 (a) and (b) show an

Fig. 1 Block-LDPC codes: (a) 4×8 base matrix $\mathbf{H}$; (b) Tanner graph corresponding to $\mathbf{H}$; (c) Block-LDPC codes generated from $\mathbf{H}$; (d) 6×6 weight-2 circulant sub-matrix.

example of 4×8 parity check matrix and its corresponding Tanner graph, respectively.

Block-LDPC codes were proposed recently almost in all of modern wireless communication systems including but not limited to IEEE 802.11n, IEEE 802.16e, DVB-S2, and CMMB. The parity check matrix $\mathbf{H}$ of these codes can be partitioned block-wise and implemented with partially parallel decoder architectures, which can achieve a good tradeoff between hardware complexity and decoding throughput. The parity check matrix $\mathbf{H}$ of Block-LDPC codes ($M = zM_b$, $N = zN_b$) is constructed from a $M_b \times N_b$ base matrix $\mathbf{H}_b$ by replacing each 1 in $\mathbf{H}_b$ with a $z \times z$ permutation sub-matrix and each 0 with a $z \times z$ zero sub-matrix, where z is an expansion factor. Figure 1 (c) shows an example of Block-LDPC code generated from base matrix in Fig. 1 (a).

In general, all nonzero sub-matrices in Block-LDPC codes are cyclically shifted identity matrices (weight-1 circulant matrices). We call this class of codes as structured Block-LDPC codes (including AA-LDPC [13], QC-LDPC [19], etc.). While a small portion of nonzero sub-matrices are compound circulant matrices (superimposed sub-matrices), we call this class of codes as unstructured Block-LDPC codes (including HS-LDPC, EG-LDPC, IRA-LDPC, etc.). As shown in Fig. 1 (d), a weight-2 circulant matrix consists of two superimposed cyclically shifted identity matrices. A general case is to have multiple (e.g., weight is bigger than 2) cyclically shifted identity matrices superimposed in one sub-matrix. As a result, the unstructured Block-LDPC codes have higher code rate and excellent error correction performance than structured Block-LDPC codes. However the irregular structure will lead to data-collision when unstructured Block-LDPC codes are decoded by TDMP algorithm, which will be explained in Sect. 2.2. Table 1 summarizes the code parameters for $\mathbf{H}$ in several standards.

Table 1 Code parameters for **H** in several standards.

LDPC Codes	WLAN	WiMAX	DVB-S2	DTMB	CMMB
M_b	4-12	4-12	45-162	11,23,35	18,36
N_b	24	24	18-135	59	72
z	27:27:81	24:4:96	360	127	128
Structured	Yes	Yes	No	Yes	No
Total modes	12	114	22	3	2

2.2 Decoding Algorithm

Most typical LDPC decoding algorithm, known as Gallager's TPMP algorithm, passes messages along the edges of the Tanner graph in multiple rounds of updates between the two classes of nodes as shown in Fig. 1 (b). Since the sequential operation and flexibility between two updates, TPMP algorithm can be rearranged into many modified schedule [6]. To make a tradeoff between decoding performance and hardware complexity, a Modified Min-Sum (MSA) algorithm based on Rearranged Schedule TPMP algorithm (RS-TPMP) [31] is adopted to decode a $zM_b \times zN_b$ Block-LDPC codes and expressed as follows:

1) Initialization:

$$R_{cv}^{(0)} = 0, S_n^{(0)} = I_n = 2y_n/\sigma^2 \quad (1)$$

2) For $i = 1$ to i_{max} iteration

For $j = 1$ to M_b , each check node c

$$\begin{aligned} L_{nc}^{(i-1)} &= S_n^{(i-1)} - R_{cn}^{(i-1)} \\ R_{cv}^{(i)} &= MSA_{n \in N(c)/v}(L_{nc}^{(i-1)}) \end{aligned} \quad (2)$$

End

For $k = 1$ to N_b , each variable node v

$$S_v^{(i)} = I_v + \sum_{m \in M(v)} R_{mv}^{(i)} \quad (3)$$

End

End

3) Check stop criterion after each iteration:

$$i > i_{max} \text{ or } H \cdot x^T = 0 \quad (4)$$

In recent years, TDMP, also known as layered belief propagation algorithm, has been proposed for decoding structured Block-LDPC codes. In TDMP, Block-LDPC codes with M_b block rows can be viewed as concatenation of M_b layers and the decoding schedule is performed block row by block row. A Modified Min-Sum Algorithm (MSA) based on TDMP algorithm can be described as follows:

1) Initialization:

$$\Lambda_{cv}^{(0)} = 0, \gamma_n^{(0)} = I_n = 2y_n/\sigma^2 \quad (5)$$

2) For $i = 1$ to i_{max} iteration

For $j = 1$ to M_b , each check node c

$$\begin{aligned} \rho_{vc} &= \gamma_n^{i(j-1)} - \Lambda_{cn}^{(i-1)} \\ \Lambda_{cv}^i &= MSA_{n \in N(c)/v}(\rho_{vc}) \\ \gamma_n^{ij} &= \rho_{vc} + \Lambda_{cv}^i \end{aligned} \quad (6)$$

End

End

3) Check stop criterion after each iteration:

$$i > i_{max} \text{ or } H \cdot x^T = 0 \quad (7)$$

Where $N(c) \setminus v$ is the set of the neighboring variable nodes for check node c excluding variable node v . $M(v)$ is the set of the neighboring check nodes for variable node v . The superscript i and j are used to indicate the i -th iteration and j -th sub-iteration, respectively. $2y_n/\sigma^2$ is the intrinsic messages related to the received soft symbol y_n and the estimated standard deviation of the channel noise. The MSA denotes the function of Modified Min-Sum algorithm. For simplicity in this paper, I_n , R_{cv} (Λ_{cv}), L_{vc} (ρ_{vc}) and S_n (γ_n) are called the value of channel intrinsic message, extrinsic message, prior message and posterior message, respectively. Corresponding to Eq. (2) and Eq. (3), the arithmetical computations in each iteration of RS-TPMP algorithm include subtraction, minimum-search and accumulation, respectively. Similarly, corresponding equation to (5) and Eq. (6), the arithmetical computations in each iteration of TDMP algorithm include subtraction, minimum-search and addition, respectively. The iteration of two algorithms will stop when the hard decision code-word x^T is satisfied to parity check equation or the iteration number i exceeds a predefined maximum number i_{max} according to Eq. (4) or Eq. (7).

TDMP algorithm achieves about two times faster decoding convergence compared to TPMP algorithm for extrinsic messages and posterior messages are updated in each block row concurrently. However, this novel decoding algorithm is not suitable for unstructured Block-LDPC codes. For example, when processing a weight-2 sub-matrix in Fig. 1 (d) in one block row of unstructured Block-LDPC codes, two clusters of γ_n are calculated out according to Eq. (6) and are immediately passed to the next block row, resulting in a data-collision in posterior messages. To solve this problem, authors in [24] have proposed a method by reordering matrix to reduce the number of data-collision in TDMP algorithm. However, this method is at cost of loss in performance for remaining conflicts and the loss of throughput for limited processing parallelism factor. So we conclude that TDMP algorithm is more suitable for being applied to QC-LDPC codes or other structured Block-LDPC codes which only consist of weight-1 circulant sub-matrix and zero sub-matrix.

Be different from TDMP algorithm, since the check-node update and the variable-node update are processed sequentially, TPMP algorithm is more flexible in each update round. To resolve the data-collision depicted above, authors in [14] take advantage of flexibility in TPMP algorithm to design a partially parallel architecture with parallel processing units and be suited for any Block-LDPC codes by adding extra processing units. Based on this method, to design a serial architecture supporting multiple code rates, we deploy a method of sub-matrix dispatch to avoid data-collision when processing superimposed sub-matrix as expressed in Fig. 2. While each block row is processed in unstructured Block-

Fig. 2 An example of sub-matrix dispatch.

LDPC codes, weight-2 circulant matrix can be composed into two weight-1 sub-matrices. According to (2) and (3), two clusters of extrinsic messages R_{cv} are calculated out, and then are accumulated according to (4) in the following block column update stage.

2.3 Combination of Two Decoding Algorithms

Differences between TPMP and TDMP have been compared both in algorithm level [7] and in hardware architecture level [20]. However, the feasibility of one LDPC decoder architecture could support two decoding algorithms has not been mentioned. In this section, we firstly address the differences in algorithm level and then rearrange the decoding schedule in two algorithms. The difference in hardware architecture level will be described in Sect. 3.

In order to efficiently decode in either TPMP mode or TDMP mode according to the structure of Block-LDPC codes, we adopt RS-TPMP algorithm and process each block row/column sequentially. The breakup of parity-check matrix partitions a check node set C into M_b disjoint subsets: $C = C_1 \cup C_2 \cup \dots \cup C_m \dots \cup C_{M_b}$, where C_m ($m = 1, 2, \dots, M_b$) denotes z check nodes in m -th block row, and variable node set V into N_b disjoint subsets: $V = V_1 \cup V_2 \cup \dots \cup V_n \dots \cup V_{N_b}$, where V_n ($n = 1, 2, \dots, N_b$) denotes z variable nodes in n -th block column. In the i -th iteration, the RS-TPMP algorithm first computes the extrinsic messages R_{cv} from V to C_1 , then V to C_2 , and so forth block row by block row. After the M_b -th block row has been processed, the decoding schedule goes into the following block column update stage and accumulates the extrinsic messages R_{cv} from C to V_1 , then C to V_2 , and so forth block column by block column as shown in Fig. 3 (a). Similarly, in the i -th iteration, the TDMP algorithm first computes the extrinsic messages Λ_{cv} from V to C_1 , and then computes posterior messages γ_n from C_1 to V , respectively. After the first sub-iteration finished, the decoding schedule goes into the second sub-iteration and computes the extrinsic messages Λ_{cv} from V to C_2 , and then computes posterior messages γ_n from C_2 to V , and so forth block row by block row as shown in Fig. 3 (b).

To reduce the complexity of hardware implementation for check-node operation, MSA algorithm is adopted both in RS-TPMP algorithm and TDMP algorithm. So there only needs minimum-search and subtraction/addition computations in place of high complexity multiplication computa-

Fig. 3 The i -th iteration of decoding schedule: (a) RS-TPMP; (b) TDMP.

Table 2 Computation comparison between RS-TPMP and TDMP.

Computation	RS-TPMP		TDMP
	Row	Column	
Cyclic Shift	$r \times M_b$	$r \times M_b$	$2 \times r \times M_b$
Data Recovery	$z \times r \times M_b$	$z \times r \times M_b$	$2 \times z \times r \times M_b$
Subtraction/Addition	$z \times r \times M_b$	$z \times r \times M_b$	$2 \times z \times r \times M_b$
Minimum Search	$z \times M_b$	0	$z \times M_b$

tions in all check-node update operations. After check-node update operations, all extrinsic messages R_{cv} (Λ_{cv}) can be stored in a compressed form with four elements [31], i.e., 1) the minimum message; 2) the second minimum message; 3) the position of the minimum message; 4) the sign of all updated extrinsic messages R_{cv} (Λ_{cv}). To recover the previous extrinsic message R_{cv} (Λ_{cv}) from this compressed form, data recovery computations are required before all check-node update operations. As a partially parallel architecture will be implemented to decode any Block-LDPC codes, cyclic shift computations are also required to cyclically permute data according to structure of sub-matrix in Block-LDPC codes. Table 2 summarizes the comparison of computations in one iteration between RS-TPMP algorithm and TDMP algorithm when decoding a $zM_b \times zN_b$ Block-LDPC code with a constant row weight is r .

3. VLSI Architecture

This section presents the developed partially parallel LDPC decoder architecture based on the analysis of previous sections. As shown in Fig. 4, our proposed architecture consists of four parts: 1) Central Controller with Permutation Memory and Address Memory; 2) Single-port memory banks including Bit Node Sum Memory (BNSM), Check Node Sum Memory (CNSM) and Temporary Data Memory (TDM), which will be detailed described in Sect. 3.2; 3) Hard Decision block, Parity Check block, Input and Output Buffer; 4) Reconfigurable Serial Processing Engine (RSPE), which

Fig. 4 Block diagram of proposed LDPC decoder.

is the central processing unit responsible for updating messages and will be detailed depicted in the Sect. 3.3.

3.1 Flexible Decoding Scheduling

To support multiple code rate and code length and to decode any Block-LDPC codes, we adopt a parameterized design and summarize parameters as follows: 1) Each of Block-LDPC codes can be represented by the location of non-zero sub-matrix and its shift size, whose value are stored in Address Memory and Permutation Memory, respectively. Especially, when decoding an unstructured Block-LDPC code which has weight-2 sub-matrix as depicted in Fig. 1 (d), we can partition this sub-matrix into two weight-1 sub-matrices. Each of two weight-1 sub-matrices has same value of location and different shift size; 2) Furthermore, other parameters such as TPMP/TDMP decoding mode, maximum iteration number, value of code-length and the size of sub-matrix can be programmed into inner control registers in Central Controller; 3) Since the parallelism factor z varies from code to code according to size of sub-matrix, the data-path has to be scalable and reconfigurable. This is achieved by employing distributed arithmetic units in RSPE and memory banks as shown in Fig. 4, resulting in a reduction of overall power consumption by deactivating the arithmetic units and memory banks that are not being used. All of the parameters above can be configured through an external configurable interface before each code-word is decoded.

The proposed decoder can be programmed into different decoding mode according to different structure of Block-LDPC codes as following criterions: when decoding a structured Block-LDPC code, the decoder can operate in TDMP decoding mode to achieve a high throughput and in TPMP decoding mode when decoding an unstructured Block-LDPC code. The decoding flow of TPMP/TDMP decoding mode can be depicted in Fig. 5 (a) and detailed described as follows:

- 1) Initialization: based on (1) and (5), BNSM and CNSM are initialized with intrinsic messages I_n and zero in both decoding mode, respectively. TDM are initialized with intrinsic messages I_n in TPMP decoding mode which will be used in block column update stage in each iteration.
- 2) Iterative decoding: a) In TPMP mode, iteration can be di-

(a)

(b)

Fig. 5 Flexible decoding Scheduling: (a) pipelined decoding process; (b) decoding flow of TPMP/TDMP.

vided into block row update stage and block column update stage. In block row update stage, the compressed extrinsic messages and the posterior extrinsic messages are firstly fetched out from CNSM and BNSM and latched in RSPE, respectively. Then RSPE recovers the extrinsic messages, cyclically shifts the posterior messages and executes subtraction and minimum-search operations to generate new compressed extrinsic messages. Finally, these new compressed extrinsic messages are written back into CNSM. After all block rows are updated, the decoding flow goes into the following block column update stage immediately. The intrinsic messages and the new compressed extrinsic messages are firstly fetched out from CNSM and TDM and latched in RSPE, respectively. Then RSPE recovers, cyclically shifts and accumulates the extrinsic messages with intrinsic messages to generate new posterior messages, finally these new messages are written back into BNSM. The iteration will terminate after all block columns have been updated; b) In TDMP mode, each of iteration can be divided into M_b successive sub-iterations according to Eq. (6). In each sub-iteration, the extrinsic messages are updated as the same operations in TPMP mode, however, the new posterior messages are generated by adding new extrinsic messages Λ_{cv} with the soft messages ρ_{vc} , which are generated from subtraction operations and buffered in TDM. These new posterior messages will be used in the next sub-iteration. The iteration will terminate after M_b sub-iterations have been finished.

- 3) Decoding output: the decoding iteration will stop and output the decoded code-word if the conditions are satisfied in Eq. (4) or Eq. (7). Otherwise, the decoding flow will go to 2) and enter into the next iteration.

In order to decode at full speed, pipelined decoding scheme is adopted by deploying Input Buffer, Output Buffer and two Hard Decision Memory blocks in our proposed ar-

chitecture, as described in Fig. 5 (b). The Input Buffer receives the intrinsic messages from outer channel and the Output Buffer outputs the decoded code-word from Hard Decision memory blocks. Two Hard Decision memory blocks are stored decoded code-word by hard decision from exclusive iteration, that is to say, one for odd number of iteration and one for even number of iteration.

3.2 Efficient Memory Implementation

Many methods have been introduced to lower the complexity and power consumption in LDPC decoder. Meanwhile, as memory occupies a large proportion both in area size and power consumption in LDPC decoder, we deploy following techniques to implement an area and power efficient memory architecture with an aim to achieve maximum throughput:

1) Memory Size Reduction: a) Modified Min-Sum algorithm (MSA) [16] and a compressed form of extrinsic messages with four elements (the first minimum message, the second minimum message, the position of the first minimum, the sign of all updated message) [17] are adopted in Eq. (2) and Eq. (6) to reduce the memory requirement for extrinsic messages; b) By rearranging schedule of TPMP algorithm, one bit node sum memory is saved compared to traditional TPMP algorithm; d) Only single-port memory is required by processing each sub-matrix sequentially in each block row/column and buffering data in register, resulting in a area reduction when compared to dual-port memory [31].

2) Memory Sharing: To maximize the hardware utilization efficiency (HUE) of memory and to store messages from different decoding mode, a memory-sharing scheme has been deployed in our proposed architecture. As depicted in Table 3, when decoding in TPMP mode, intrinsic messages I_n , extrinsic messages R_{cv} and posterior messages S_n are stored in TDM, CNSM and BNSM, respectively. While decoding in TDMP mode, soft messages ρ_{vc} , extrinsic messages Λ_{cv} and posterior messages γ_n are stored in TDM, CNSM and BNSM, respectively.

3) Memory Power Reduction: a) TDM, BNSM and CNSM are implemented by employing distributed memory banks which are composed of compact register files as shown in Fig. 4 and the unused register file banks can be deactivated when decoding, resulting in a reduction of overall power consumption; b) Less temporary data have to be stored since the reduction of memory size, resulting in a less number of memory access operation when compared to traditional decoding algorithms; c) By buffering data fetched from memory, such as the compressed extrinsic messages in each block row update stage, less number of read access operation is needed. These two methods can reduce the power consumption by reducing the number of memory access operation which is most dominant power-consumption factor in memory-based LDPC decoder architectures.

To verify our proposed efficient memory implementation scheme, a flexible LDPC decoder which supports WiMAX standard has been implemented and will be de-

Table 3 Memory sharing scheme.

Decoding Mode	Temporary Data Memory (TDM)	Check Node Sum Memory (CNSM)	Bit Node Sum Memory (BNSM)
RS-TPMP	I_n	R_{cv}	S_n
TDMP	ρ_{vc}	Λ_{cv}	γ_n

Fig. 6 Memory structure of the proposed LDPC decoder on WiMAX standard.

tailed depicted in Sect. 4. Figure 6 presents the memory structure of this novel LDPC decoder.

3.3 Reconfigurable Serial Processing Engine

In partially parallel architecture, the parallelism factor is equal to the size of sub-matrix (expansion factor z), where the value varied from code to code. To design a data-path that is regular and scalable to support different code type, a Reconfigurable Serial Processing Engine (RSPE) with maximum parallelism factor of z_{max} has been proposed as shown in Fig. 4, where the unused processing units (when $z < z_{max}$) can be deactivated to reduce the overall power consumption. The RSPE is implemented as the following methods.

As introduced in Sect. 2.2, the decoding flow of TPMP algorithm can be divided into the check-node update stage and variable-node update stage and each stage is performed sequentially, resulting in nearly 50% of hardware utilization efficiency (HUE) of processing units because all the VNUs are idle while the CNUs are busy during the check-node update stage and *vice versa* during the variable-node update stage as in [10], [11] and [14]. Similar operations are performed when processing check-node and variable-node in TDMP algorithm. To increase the HUE of processing units or the throughput of LDPC decoder, phase-overlapping message passing schemes have been proposed in [10] and [11]. However, only limited HUE or throughput has been achieved because these methods are code dependent. Furthermore, as described in Table 2, only four kinds

Fig. 7 The proposed Reconfigurable Serial Processing Engine (RSPE).

of basic computation are required in RS-TPMP algorithms or TDMP algorithms. Correspondingly, there are four basic arithmetic processing units required in our proposed RSPE as shown in Fig. 7: a) Data Recovery Array (DRA), which recovers the extrinsic messages from the compressed form fetched out from CNSM; b) Reconfigurable Cyclic Shifter (RCS), which is realized with based on [21] and [22]; c) Finite Addition Array (FAA), which achieves the signed binary addition computations with overflow controll; d) Minimum Value Search Array (MVSA), which searches out the minimum message, the minimum message, the position of the minimum message and the sign of all updated extrinsic message in each check-node update. In order to increase the HUE, Fig. 7 presents a novel RSPE which is composed of two RCSs, a DRA, a FAA, a MVSA and three 2-to-1 MUXs. Each array of processing units can be set into activated/deactivated and three 2-to-1 MUXs could switch data-path according to different update stage. As shown in Fig. 8(a) and (b), when the proposed decoder is decoding in TPMP mode, RSPE can operate in R_{cv} update stage and S_n update stage, respectively. Similarly, when decoding in TDMP mode, RSPE can operate in Λ_{cv} update stage and γ_n update stage as shown in Fig. 8(c) and (d), respectively. Table 4 summarizes the operating conditions in different update stages.

It can be observed that both TPMP algorithm and TDMP algorithm have unbalanced computation complexity between check-node update stage and variable-node update stage. On the other hand, as the maximum operating frequency will be upper-bounded by the critical path in processing units, the throughput of LDPC decoder will be limited. In order to reduce the critical path and balance the computation complexity in processing units, a six-stage (including memory write and read stages) pipeline structure partitioned by functional blocks is adopted. As shown in Fig. 7, D latches are inserted into data-path between two adjacent functional processing units. As a result, the decoding throughput is increased to a certain extent to meet the demands of system.

In partially parallel architecture, the interconnection network that is responsible for routing messages between block rows and block columns should be configurable to implement a flexible LDPC decoder to support multiple code-rate and code-length. Based on the concept of programmable shuffle network introduced in [21] and [22], a Reconfigurable Cyclic Shifter (RCS) whose data-path can

Fig. 8 Different decoding stages: (a) R_{cv} update stage in RS-TPMP; (b) S_n update stage in RS-TPMP; (c) Λ_{cv} update stage in TDMP; (d) γ_n update stage in TDMP.

be bypassed is proposed with permutation size p and expansion factor z , where permutation size p is stored in Permutation Memory and expansion factor z is configured in inner registers through external configurable interface. In our proposed RSPE, two RCSs are employed to get a trade-off between routing complexity and area, which can be configured into different operating mode as shown in Fig. 7.

Table 4 Operating conditions of units. Note: 1) the number ‘0’ indicates the selection condition of MUX; 2) ‘on/off’ indicates the processing units are active/deactivated.

Computation	RS-TPMP		TDMP	
	R_{cv} update	S_n update	R_{cv} update	S_n update
MUX 1	1	0	0	0
MUX 2	0	0	0	1
MUX 3	1	1/0	1	1
DRA	on	on	on	on
FAA	on	on	on	on
MUSA	on	off	on	off
RCS 1	on	off	on	on
RCS 2	off	on	off	on

4. Chip Measurement

To verify our proposed architecture and be convenient to compare with the state-of-the-art LDPC decoders of [10], [11] and [22], we have implemented a decoder on IEEE 802.16e standard which has multiple code-rate and code-length. According to systematic simulation, for representation of channel intrinsic messages, a bit-width of 6 bit, with 4 bit for the integer part and 2 bit for the fractional part, is sufficient. Figure 9 shows the fixed-point simulation results when using RS-TPMP algorithm and TDMP algorithm with different decoding iteration number for rate-1/2 and 2304-bit code. As depicted in Sect. III. A, the maximum iteration number can be set by an external configurable interface according to the channel condition. When decoding in TPMP (TDMP) mode, the maximum iteration number is set to 30 (20) to get a tradeoff between throughput and error correction performance.

There are 114 modes (6 code rates, each code rate has 19 code lengths) defined in the IEEE 802.16e system, where the number of block column (N_b) in each code rate is equally 24, the maximum number (M_b) of block row in 6 code rates is 12, and the 19 variable expansion factors (z) corresponding to 19 code lengths range from 24 to 96 with an increment of 4. Therefore, a RSPE with maximum parallelism factor of 96 is deployed. The depths of DTM, CNSM and BNSM are 24, 12 and 24, respectively. The locations and shift sizes of sub-matrices corresponding to each parity-check matrix $\mathbf{H}$ are programmed into Address Memory and Permutation Memory through the external configurable interface at the initialization stage of decoding, respectively.

The net throughput of our proposed LDPC decoder can be estimated as the following equation:

$$\text{Throughput} = \frac{N \times R \times f_c}{T_{ini} + T_{ite} \times N_i} \quad (8)$$

Where N , R and f_c denote the code length ($N = z \times N_b$), the code rate and the operating clock frequency, respectively. The first part T_{ini} of the denominator represents the clock cycles for initialization, and the second part $T_{ite} \times N_i$ represents the clock cycles for iterative decoding, where T_{ite}

(a)

(b)

Fig. 9 Performance of fixed point simulation at rate-1/2 2304-bits code: (a) RS-TPMP algorithm; (b) TDMP algorithm.

and N_i denote the clock cycles of one iteration and the iteration number. Based on different combinations of operating frequency f_c , code-rate R and code-length N , the net throughput of the implemented decoder can be tuned from 1.93 Mb/s to 2.76 Gb/s. The value 1.93 Mb/s is achieved at an operating frequency of 50 MHz, when decoding a rate-1/2 576-bit code with an iteration number 30. And the value 2.76 Gb/s is achieved at an operating frequency of 250 MHz, when decoding a rate-5/6 2304-bit code with an iteration number one.

To compare with other decoders, we define Normalized Energy Efficiency (NEE) to evaluate the decoders implemented with different fabrication technologies (similar to [23]).

NEE can be estimated as the following equation:

$$NEE = \frac{P \times R}{\text{Throughput} \times N_i} \times \left(\frac{130 \text{ nm}}{\text{Technology } y} \right)^2 \quad (9)$$

Where P denotes the power consumption, NEE describe the energy required to decode one bit of an LDPC code block with 130 nm CMOS technology and once of iteration.

Table 5 Features of the LDPC decoder in IEEE 802.16e.

	The Proposed Decoder
Code Length	576:96:2304
Code Rate	1/2, 2/3A, 2/3B, 3/4A, 3/4B, 5/6
Expansion Factor	24:4:96
Parity Check Matrix	QC-LDPC
Logic Gates	215 K
Total Memory	78,336 bits (single-port register files)
Maximum Frequency	250 MHz
Die Area	6.3 mm ²
Power	592 mW @ 1.2dB, 250 MHz
Throughput	1.93 Mb/s ~ 2.76 Gb/s
IO pins	80 Pin QFP
Technology	Standard 130nm 1P8M CMOS 1.2V Core 2.5V IO

Fig. 10 Die photo of the proposed LDPC decoder.

Table 5 summarizes the chip implementation results of the proposed LDPC decoder. The decoder chip occupies an area of 6.3 mm² with 215 K logic gates and 156 K bits memory and achieves a maximum decoding net throughput of 2.76 Gb/s at an operating frequency of 250 MHz with maximum one iteration when decoding in TDMP decoding mode. The power consumption is 592 mW when decoding at 250 MHz and 1.2 V supply voltage when decoding in TDMP decoding mode. The die micrograph of the chip is shown in Fig. 10 and Table 6 compares this decoder with the state-of-the-art LDPC decoders of [10], [11], [13] and [22].

5. Conclusion

In this paper, a flexible LDPC decoder architecture which is code independent has been proposed. This novel architecture can be configured to decode in TDMP mode to maximize throughput when decoding structured Block-LDPC codes such as QC-LDPC codes, or in TPMP mode to achieve flexibility when decoding unstructured Block-

Table 6 Comparison with the existing LDPC decoders. The features^{(1)–(2)} are given by Eq. (8) and Eq. (9) for ease of comparison with other decoders.

	[10]	[11]	Proposed
Chip Area (mm ²)	8.29	6.25	6.3
Gates (K)	420 (Total)	380 (Logic) 590 (Memory)	215 (Logic) 156 (Memory)
Memory Bits	76,800	89,856	78,336
Parallelism	4 CNUs, 8 BNUs	192 CNUs, 96 BNUs	96 RSPE
Net Throughput (Mb/s)@(frequency, N _t , coderate)	222.2 @(83.3, 2, N/A)	105 ⁽¹⁾ @(150, 20, 5/6)	276 @(250, 10, 5/6)
Decoding Algorithm	TPMP	TPMP	RS-TPMP, TDMP
Frequency (MHz)	83.3	150	250
Power (mW) @(SNR, Frequency) or (Throughput)	52 @(N/A, 83.3)	264 @(105Mb/s)	592 @(1.2dB, 250)
Energy Efficiency	N/A	208 (125× (5/6) ×2) ⁽²⁾	256
Operating Mode	114	114	114
CMOS Technology	130nm, 1.2V	90nm, 1.0V	130nm, 1.2V

LDPC codes. As a result, the proposed decoder essentially combines the advantages of two decoding algorithms. Meanwhile, compared with other state-of-the-art LDPC decoders which support multiple code rates and code lengths, the proposed architecture features high area efficiency and has a scalable data-path, which is particularly suitable for multi-standard LDPC decoder scheme.

References

- [1] R. Gallager, "Low-density parity-check codes," IEEE Trans. Inf. Theory, vol.IT-8, pp.21–28, Jan. 1962.
- [2] B. Bangerter, E. Jacobsen, M. Ho, A. Stephens, A. Maltsev, A. Rubtsov, and A. Sadri, "High-throughput wireless LAN air interface," Intel Technol. J., vol.7, no.3, Aug. 2003. [Online] Available: <http://www.intel.com/technology/itj>
- [3] IEEE802.16e. [Online]. Available: <http://www.ieee802.org/16/tge/>
- [4] Digital Video Broadcasting (DVB-S2) Via Satellite [Online]. Available: <http://www.dvb.org>
- [5] Mobile Multimedia Broadcasting (P.R. China) Part 1: Framing Structure, Channel Coding and Modulation for Broadcasting Channel [S]. Beijing: State Administration of Radio, Film and Television, 2006.
- [6] F. Guilloud, E. Boutillon, J. Tusch, and J.-L. Danger, "Generic description and synthesis of LDPC decoders," IEEE Trans. Commun., vol.55, no.11, pp.2084–2091, Nov. 2007.
- [7] M. Mansour, "A turbo-decoding message-passing algorithm for sparse parity-check codes," IEEE Trans. Signal Process., vol.54, no.11, pp.4376–4392, Nov. 2006.
- [8] H. Zhang and T. Zhang, "Block-LDPC: A practical LDPC coding system design approach," IEEE Trans. Circuits Syst. I, vol.52, no.4, pp.766–775, April 2005.
- [9] A.J. Blanksby and C.J. Howland, "A 690 mW 1 Gb/s 1024b rate 1/2 low density parity check code decoder," IEEE J. Solid-State Circuit, vol.37, no.3, pp.404–412, March 2002.
- [10] X.-Y. Shih, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "An 8.29 mm² 52 mW multi-mode LDPC decoder design for mobile WiMAX sys-

- tem in 0.13 μm CMOS process," IEEE J. Solid-State Circuit, vol.43, no.3, pp.672–683, March 2008.
- [11] 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 Circuit, vol.44, no.3, pp.684–694, March 2008.
 - [12] Y. Sun and J.R. Cavallaro, "A low-power 1-Gbps reconfigurable LDPC decoder design for multiple 4G wireless standards," IEEE Int. SOC Conf., pp.367–370, Sept. 2008.
 - [13] M.M. Mansour and N.R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," IEEE J. Solid-State Circuit, vol.41, pp.684–698, March 2006.
 - [14] Z.-F. Wang and Z.-Q. Cui, "Low-complexity high-speed decoder design for quasi-cyclic LDPC codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.15, no.1, pp.104–114, Jan. 2007.
 - [15] S.M.E. Hosseini, K.S. Chan, and W.L. Goh, "A reconfigurable FPGA implementation of an LDPC decoder for unstructured codes," 2nd Int. Conf. Signals, Circuits and Systems, pp.1–6, Nov. 2008.
 - [16] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," IEEE Trans. Commun., vol.50, no.3, pp.406–414, March 2002.
 - [17] H. Zhong, W. Xu, N.-D. 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.
 - [18] D. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol.45, no.2, pp.399–431, March 1999.
 - [19] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-limit quasi-cyclic low-density parity-check codes," IEEE Trans. Commun., vol.52, no.7, pp.1038–1042, July 2004.
 - [20] B. Torben, A. Matthias, L.-E. Timo, K. Frank, W. Norbert, B. Friedbert, and R. Andreas, "A survey on LDPC codes and decoders for OFDM-based UWB systems," Vehicular Technology Conf., pp.1549–1553, April 2007.
 - [21] Y.-S. Deng, "Reconfigurable LDPC decoder for digital video broadcasting systems," M.S. Dissertation, State Key Lab. Of ASIC and Systems, Fudan University, April 2008.
 - [22] D. Bao, B. Xiang, R. Shen, A. Pan, Y. Chen, and X.-Y. Zeng, "Programmable architecture for flexi-mode QC-LDPC decoder supporting wireless LAN/MAN applications and beyond," IEEE Trans. Circuits Syst. I, vol.57, no.1, pp.125–138, Jan. 2010.
 - [23] H.-Y. Hsu, J.-C. Yeo, and A.-Y. Wu, "Multi-symbol-sliced dynamically reconfigurable Reed-Solomon decoder design based on unified finite-field processing element," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.14, no.5, pp.489–500, May 2006.
 - [24] C. Marchand, J.-B. Dore, L. Conde-Canencia, and E. Boutillon, "Conflict resolution for pipelined layered LDPC decoders," IEEE Workshop on Signal Processing Systems, pp.220–225, Nov. 2009.
 - [25] J. Dielissen, A. Hekstra, and V. Berg, "Low cost LDPC decoder for DVB-S2," Proc. Design, Automation and Test in Europe (DATE'06), vol.2, pp.1–6, Munich, Germany, March 2006.
 - [26] G. Gentile, M. Rovini, and L. Fanucci, "Low-complexity architectures of a decoder for IEEE 802.16e LDPC codes," Euromicro Conf. on Digital System Design Architecture, Methods and Tools (DSD'07), pp.369–375, Lubeck, Germany, Aug. 2007.
 - [27] P. Urard, L. Paumier, V. Heinrich, N. Raina, and N. Chawla, "A 360 mW 105 Mb/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes enabling satellite-transmission portable devices," IEEE Int. Solid-State Circuit Conf. (ISSCC) Dig. Tech. Papers, pp.310–311, Feb. 2008.
 - [28] C. Zhang, Z.-F. Wang, S. Jin, L. Li, and L. Jun, "Flexible LDPC decoder design for multigigabit-per-second applications," IEEE Trans. Circuits Syst. I, vol.57, no.1, pp.116–124, Jan. 2010.
 - [29] Y.-Y. Lee and H.-J. Ryu, "A 1-Gb/s flexible LDPC decoder supporting multiple code rates and block lengths," IEEE Trans. Consum. Electron., vol.54, no.2, pp.417–424, May 2008.
 - [30] S. Kim, G.-E. Sobelman, and H. Lee, "Flexible LDPC decoder architecture for high-throughput applications," Proc. IEEE APCCAS, vol.1, pp.45–48, Macao, P.R. China, Nov. 2008.
 - [31] B. Xiang, R. Shen, A. Pan, D. Bao, and X.-Y. Zeng, "An area-efficient and Low-power multirate decoder for quasi-cyclic low-density parity-check codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.18, no.99, p.1, Aug. 2009.
 - [32] M. Alles, T. Vogt, and N. When, "FlexiChap: A reconfigurable ASIC for convolutional, turbo, and LDPC code decoding," Int. Symposium on Turbo Codes and Related Topics, pp.84–89, Sept. 2008.
 - [33] F. Vacca, G. Masera, H. Moussa, A. Baghdadi, and M. Jezequel, "Flexible architecture for LDPC decoders based on network on chip paradigm," Euromicro Conf. on Digital System Design, Architecture, Methods and Tools (DSD'09), pp.582–589, Patras, Aug. 2009.
 - [34] R. Priewasser, M. Huemer, and B. Bougard, "Trade-off analysis of decoding algorithms and architectures for multi-standard LDPC decoder," Workshop on Signal Processing Systems (SiPS 2008), pp.88–93, Washington, DC, Oct. 2008.
 - [35] Framing Structure, Channel Coding and Modulation for Digital Television Terrestrial Broadcasting system [S], R.P. China Standard GB20600-2006, 2006.

Shuangqu Huang received the B.S. degree in microelectronics from Nankai University, Tianjin, P.R. China, 2004. From Jul. 2004 to Sep. 2007, he worked as an assistant engineer with Grand Digital Information Limited Company, Beijing, P.R. China. He is currently working toward his M.S. degree in microelectronics at the State Key Lab. of ASIC & System, Fudan University, Shanghai, P.R. China. His research interests include ASIC designs, channel decoding techniques and VLSI architectures for broad-band wireless transmission systems.

Xiaoyang Zeng received the B.S. degree from Xiangtan University, China in 1992, and the Ph.D. degree from Changchun Institute of Optics and Fine Mechanics, Chinese Academy of Sciences in 2001. From 2001 to 2003, he worked as a post-doctor researcher at the State-Key Lab of ASIC and System, Fudan University, P.R. China. Then he joined the faculty of Department of Micro-electronics at Fudan University as an associate professor. His research interests include information security chip design, VLSI signal processing, and communication systems. Prof. Zeng is the Chair of Design-Contest of ASP-DAC 2004 and 2005, also the TPC member of several international conferences such as ASCON 2005 and A-SSCC 2006, etc.

Yun Chen received the B.S. and M.S. degrees in microelectronics from UESTC, China, in 2001 and 2004, respectively, and the Ph.D. degree from Fudan University in 2007. Dr. Chen is currently a lecturer with the State Key Laboratory of ASIC & System, Microelectronics Department, Fudan University, Shanghai, China. Her research interests include VLSI architectures and integrated circuit (IC) design for communications and digital signal processing systems. She serves as a member of the Technical Program Committee of the IEEE International Conference on ASIC in 2009.