

A 2.37-Gb/s 284.8 mW Rate-Compatible (491,3,6) LDPC-CC Decoder

Chih-Lung Chen, Yu-Hsiang Lin, Hsie-Chia Chang, and Chen-Yi Lee, *Member, IEEE*

Abstract—This paper presents a (491,3,6) time-varying low-density parity check convolutional code (LDPC-CC) decoder chip. This work combines the algorithm level, node level, and bit level optimizations to achieve over 2 Gb/s throughput with acceptable hardware cost and power. The algorithm level optimization is the on-demand variable node activation scheduling with concealing channel values, which can not only achieve twice faster decoding convergence speed than log-belief propagation (log-BP) algorithm, but also reduce the 17% message storage capacity. The node level optimization duplicates the check node units and variable node units and unfolds the message storage first-in–first-outs (FIFOs) so that the throughput becomes twelve multiplying with clock frequency. In the meantime, the bit level optimization is employed to retime the critical path such that the higher clock frequency can be achieved and message storage size is slightly reduced. Furthermore, a novel hybrid-partitioned FIFO is proposed to provide sufficient memory bandwidth to processing units and alleviate power consumption. With these schemes, a test chip of proposed LDPC-CC decoder has been fabricated in 90 nm CMOS technology with core area of $2.37 \times 1.14 \text{ mm}^2$. Maximum throughput 2.37 Gb/s is measured under 1.2 V supply with energy efficiency of 0.024 nJ/bit/proc. Depending on the operation mode, power can be scaled down to 90.2 mW while maintaining 1.58 Gb/s at 0.8 V supply.

Index Terms—Decoding scheduling, error correction, high throughput, low-density parity check convolutional code (LDPC-CC).

I. INTRODUCTION

LOW-DENSITY PARITY CHECK (LDPC) codes can be classified into two categories, one is LDPC block codes (LDPC-BCs) and the other is LDPC convolutional codes (LDPC-CCs). LDPC-BCs were first introduced by Gallager in early 1960s [1], but were disregarded for unfeasible architectures and VLSI technologies. MacKay and Neal rediscovered LDPC-BCs in 1997 [2] and proposed the belief-propagation (BP) decoding algorithm which successfully provided near Shannon bound error-correcting performance [3]. Simplified

decoding algorithms such as min-sum (MS) algorithm and normalized min-sum (NMS) algorithm reduced the decoder complexity with acceptable performance loss [4]. Further scheduling algorithms named as shuffled decoding and layered decoding successfully drove the overall throughput to multi-Gb/s [5], [6]. Therefore LDPC-BCs are widely adopted in many communication systems, such as IEEE802.11n, IEEE802.15.3c, IEEE 802.16e, and DVB-S2 [7]–[10].

Near the rediscovery of LDPC-BCs, LDPC-CCs were proposed in 1999 [11]. LDPC-CCs have the characteristics of convolutional code not found in LDPC-BCs. Continuous encoding supports any length of input data stream, which is especially suitable for streaming video and packet-switching network. The puncture scheme applied in LDPC-CCs provides flexible code-rates by abandoning certain positions of encoded bits according to the puncture table. Simple encoder circuitry composed by registers, multiplexers, and a few XOR gates has lower hardware cost and power consumption, and can be used in distributed sensor network. Furthermore, the correlation between codeword symbols of LDPC-CCs is limited to a specific interval (constraint length $m_s + 1$, m_s is the memory size of encoder). This locality property lowers the overall routing complexity of the decoder. Although possessing many advantages, LDPC-CCs were rarely chosen by standards. The main reason lies in its bottlenecks of the long decoding latency, high power consumption, and low-to-moderate decoding throughput.

The throughput of LDPC-CC decoders reported in literatures were only several hundred Mb/s, which were difficult to compete with over Gb/s LDPC-BC decoder. Cause of lower throughput can be explained by the decoder structure. LDPC-CC decoder consists of serially concatenated processors, and each processor decoding a sliding window on the bipartite graph can be taken as one iteration in LDPC-BCs. Increasing processor number can enhance error-correcting capability but cannot increase throughput. Therefore many works put efforts on realizing the parallel message passing: analysis of parallelization concepts in [12], single-instruction–multiple-data (SIMD) architecture in [13], and joint code-decoder design in [14]. Recently, a high throughput LDPC-CC decoder design was proposed by adding regularity during code construction [15]. However, achieving high throughput is still challenging for some time-varying LDPC-CCs without regularity.

The high power consumption is another crucial issue for LDPC-CC decoder. The nature of LDPC-CC is similar to convolutional code that the received messages are continuous stream. Therefore the decoder consists of many message streams which behave like first-in–first-out (FIFO). Current literatures can be classified to two FIFO implementation

Manuscript received September 08, 2011; revised November 26, 2011; accepted December 20, 2011. Date of publication February 24, 2012; date of current version March 28, 2012. This paper was approved by Guest Editor Makoto Nagata. This work was supported by the National Science Council and Ministry of Economic Affairs of Taiwan, under Grants NSC 100-2220-E-009-062 and NSC 100-2221-E-009-044-MY3.

The authors are with the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan (e-mail: lung@si2lab.org; hcchang@mail.nctu.edu.tw; cylee@si2lab.org).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/JSSC.2012.2185193

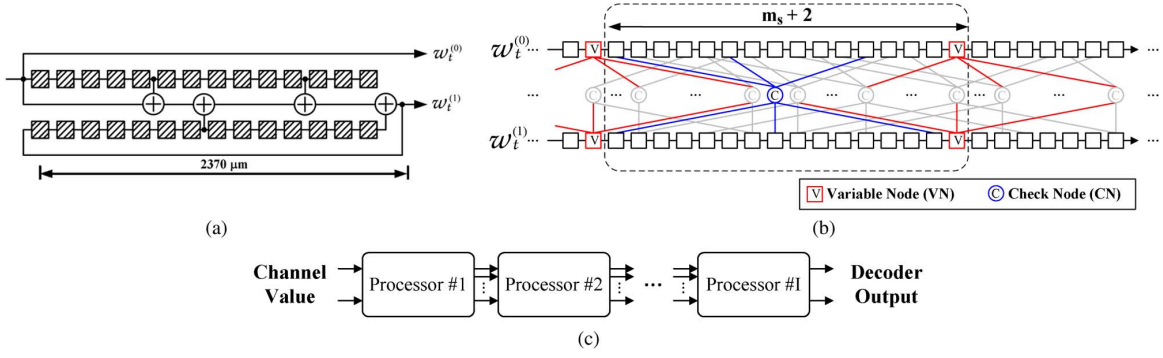


Fig. 1. (14, 3, 6) time-invariant LDPC-CC example. (a) Encoder. (b) sliding window decoding. (c) pipeline decoder.

styles, register-based [16], [17] and memory-based [18]–[22]. Register-based FIFO supports unlimited message bandwidth but introduces enormous power consumption. Contrarily memory-based FIFO reduces power consumption but suffers severe memory conflict. The limited bandwidth of memory will highly degrade the overall throughput especially with high parallelism.

In order to reduce decoding latency, increase throughput, and reduce hardware costs and power consumption, this paper proposes three levels of optimization (algorithm level, node level, bit level) and hybrid partitioned FIFO. The idea of algorithm-level optimization is inspired by the layered decoding of LDPC-BCs decoder which can accelerate the convergence speed of decoding [6]. In reference to the same error rate, layered decoding is able to reduce about half of the iteration number. Later similar idea was found to apply to the LDPC-CCs which was named as on-demand variable-node activation scheduling (abbreviated as OVA scheduling in this paper) [23], [24]. We further modified the OVA scheduling by hiding the original channel values into the streams of message passing, which not only maintain the original double convergence rate (i.e. only half processing units and decoding latency are required to reach the same error-correcting performance) but also reduce overall size of FIFOs. The node-level optimization employs register folding technique with node duplication, which increases the throughput of each processor to the product of node parallelism (ρ) and the operating frequency (f_{clk}). Bit-level optimization will further perform retiming to variable nodes, shorten the longest delay path, thereby, increase the operating frequency. In addition, after register folding each FIFO has to provide the ρ -fold bandwidth in time-invariant LDPC-CC. As for time-varying case, each FIFO has to be duplicated to ρ -fold, causing huge hardware costs and power consumption. The proposed hybrid partitioned FIFO splits out continuous section of each FIFO, and the sections with similar length are combined to one memory bank, reducing hardware costs and power consumption. The remaining parts are still implemented by registers in order to provide sufficient bandwidth for high node parallel decoder.

Finally, the proposed three levels of optimization and hybrid partitioned FIFO circuit is implemented in 90 nm process chip. The code specification is $(m_s = 491, J = 3, K = 6)$ LDPC-CC, where J and K are the degree of variable node and check node. The proposed decoder consisting of 5 processors

can achieve error-correcting performance compatible to 10 iterations of conventional algorithm. After optimization, the node parallelism equals to 12 and 50% of FIFO are split to three two-port memories. The measurement results show that the decoder can achieve 2.37 Gb/s throughput with maximum operating frequency 198 MHz and power consumption 284 mW under 1.2 V supply voltage.

The rest of this paper is organized as follows. The iterative decoding algorithm of LDPC-CC as well as its scheduling algorithm is reviewed in Section II. The improved decoding algorithm and three implementation techniques are proposed and analyzed in Section III. Moreover, Section IV reports the implementation results of (491,3,6) LDPC-CC code, including bit-error rate (BER) performance and measurement results of the 90-nm chip. Finally, the conclusions are given in Section V.

II. BACKGROUND

An LDPC convolutional code (LDPC-CC) is a convolutional code with a sparse and semi-infinite parity-check matrix $\mathbf{H}$. A parity-check matrix $\mathbf{H}$ in the scalar form can be written as

$$\mathbf{H}^T = \begin{bmatrix} \ddots & & \ddots & & \\ \mathbf{H}_{0,t-m_s}^T & \cdots & \mathbf{H}_{m_s,t}^T & & \\ & \ddots & \vdots & \ddots & \\ & & \mathbf{H}_{0,t}^T & \cdots & \mathbf{H}_{m_s,t+m_s}^T \\ & & \vdots & & \ddots \end{bmatrix} \quad (1)$$

where $\mathbf{H}_{i,t}^T$ are submatrices of size $c \times (c - b)$ with code-rate $R = b/c$ ($b < c$). For a (m_s, J, K) -regular LDPC-CC, there are exactly J ones in each row and K ones in each column of $\mathbf{H}^T$. The constraint length is defined as $\nu = c \cdot (m_s + 1)$. Normally the submatrices are periodic in time, which means $\mathbf{H}_{i,t}^T = \mathbf{H}_{i,t+T_s}^T$ for any i and t , and this kind of code is called time-varying LDPC-CC with period T_s . If $T_s = 1$, it is time-invariant since each column of $\mathbf{H}^T$ is the down-shifted version of previous column.

Assume the codeword vector of an LDPC-CC is denoted as $\mathbf{w} = [\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_t, \dots]$, where $\mathbf{w}_t = (w_t^{(0)}, \dots, w_t^{(c-1)})$, then it will satisfy the parity-check equation, i.e., $\mathbf{w}\mathbf{H}^T = 0$. Without loss of generality, in this article the code-rate R is assumed to 1/2 and the encoding scheme is assumed to systematic. Accordingly, the codeword symbol and the submatrix of $\mathbf{H}$ at each time instance are $\mathbf{w}_t = [w_t^{(0)}, w_t^{(1)}]$ and $\mathbf{H}_{i,t} =$

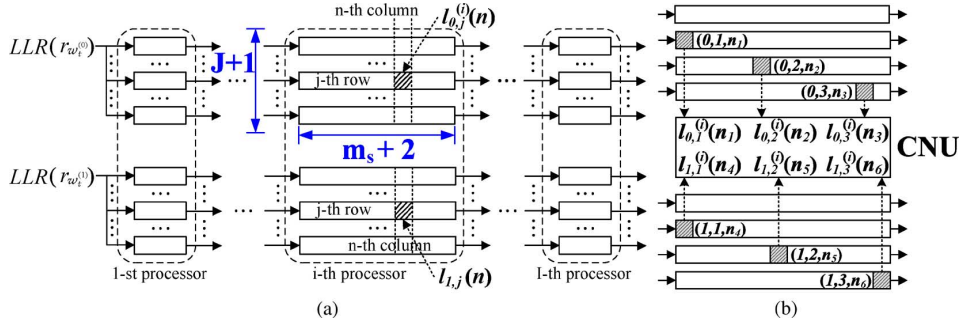


Fig. 2. Illustration of notations. (a) Illustration of indexes i , j , n , and w . (b) illustration of CNU input set.

$[h_{i,t}^{(0)}, h_{i,t}^{(1)}]$, where $w_t^{(0)}$ and $w_t^{(1)}$ are the information bit and parity bit respectively. Therefore, the parity check equation is rearranged to

$$\sum_{i=0}^{m_s} w_{t-i}^{(0)} h_{i,t}^{(0)} + \sum_{i=0}^{m_s} w_{t-i}^{(1)} h_{i,t}^{(1)} = 0. \quad (2)$$

A rate 1/2, (14, 3, 6) time-invariant LDPC-CC is given as an example. The encoder is shown in Fig. 1(a) and check equation is described by

$$w_t^{(0)} + w_{t-5}^{(0)} + w_{t-11}^{(0)} + w_t^{(1)} + w_{t-7}^{(1)} + w_{t-14}^{(1)} = 0. \quad (3)$$

A. Message-Passing Algorithm for LDPC-CC

Based on the relationship of parity-check equation, the bipartite graph is depicted in Fig. 1(b). The upper row and lower row are codeword symbols representing variable nodes (VN). Each nonzero element of $\mathbf{H}$ is mapped to one edge in the graph. The parity-check equations are transformed to check nodes (CN). Because the stream characteristics of LDPC-CC, the decoding can be carried out by sliding window decoding. Each window performs one CN operation and two VN operations. From (2), the correlation with largest distance in CN operation is $m_s + 1$. Here we assume that CN and VN operations are completed in the individual cycle, then the sliding window size is $m_s + 2$. One sliding window executing the message passing from CN to VN and VN to CN represents one iteration. The entire decoder consists I processors, where each processor implements one sliding window. The structure of pipelined decoder is illustrated in Fig. 1(c).

The decoding algorithm for LDPC-CC is the message-passing algorithm performed on the graph. The message storage scheme is in view of VNs. Since every VN has one received channel value and messages on the J connecting edges, there are totally $2(J + 1)$ messages (information plus parity) in each time instance. Combine the messages in all time instance within one sliding window to message streams, where each message stream acts like first-in first-out (FIFO). Therefore in every processor there are $2(J + 1)$ FIFOs storing streams of channel values and messages. The decoding of LDPC-CC is developed according to this message storage scheme. Let's start from the definition of index i , j , n , and w . The index i indicates i -th processor among I processors. The index j means j -th FIFO or j -th message send out from a VN. If j equals to zero, that is the FIFO storing channel values. The index n stands

for the n -th address in a FIFO or n -th VN. The last index w is used to distinguish the information part ($w_t^{(0)}$) and the parity part ($w_t^{(1)}$). By means of index definition, the resulted message storage scheme is shown in Fig. 2(a). The $l_{w,j}^{(i)}(n)$ are j -th VN to CN message send out from n -th VN node at i -th iteration, where w is either 0 for information part or 1 for parity part. On the other hand, $\varepsilon_{w,j}^{(i)}(n)$ is the CN to VN messages.

Through the definition of indexes, the message-passing decoding algorithm can be described as follows. Initially, all the shift registers in the pipeline decoder are filled with infinite because the dummy zeros are the initial values in the encoder. As the log-likelihood ratio (LLR) of received channel values $r_{w_t^{(0)}}$ and $r_{w_t^{(1)}}$ are received, they are shifted into the inputs of the $2(J + 1)$ rows of shift registers. The contents of shift registers are simultaneously shifted one time instance toward output. The next step is to operate the check node computations, and then update the variable nodes which are in the end of sliding window. Here we use the NMS algorithm, where the check node operation is calculate as

$$\varepsilon_{w,j}^{(i)}(n) \approx \prod_{(w',j',n') \in C_p \setminus (w,j,n)} \text{sign} \left(l_{w',j'}^{(i-1)}(n') \right) \times \min_{(w',j',n') \in C_p \setminus (w,j,n)} \left\{ \left| l_{w',j'}^{(i-1)}(n') \right| \right\} \times \beta \quad (4)$$

where $p = t_0 \bmod T_s$. The scaling factor β will influence the performance, therefore it needs to be chosen by simulations. C_p defines the check-node input set at the p -th phase, in which each element (w, j, n) is the VN to CN message location in FIFOs and $|C_p| = K$. Fig. 2(b) depicts the check-node input set of a time-invariant LDPC-CC with CN degree $K = 6$, where the values of n_1 to n_6 depend on the code definition.

After check node updates, variable node is computed by

$$l_{w,j}^{(i)}(m_s + 2) = l_{w,0}^{(i)}(m_s + 2) + \sum_{j' \neq j} \varepsilon_{w,j'}^{(i)}(m_s + 2). \quad (5)$$

The decoding procedure successively repeats the shifting step and appropriate node updates. As long as the initial decoding delay has elapsed total $I \times (m_s + 2)$ time units, the pipeline decoder outputs a decoded data.

According to the message-passing algorithm for LDPC-CC, the conventional processor architecture is illustrated in Fig. 3. There are only one check node unit (CNU) and two variable node units (VNUs) in a processor, where CNU and VNU compute (4) and (5), respectively. Each square indicates a CN to VN

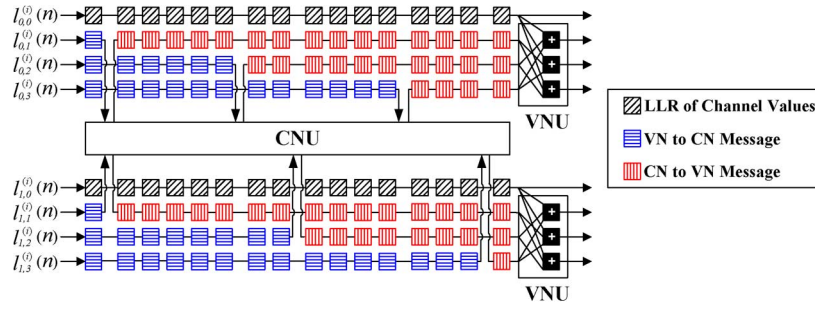


Fig. 3. Conventional processor architecture.

TABLE I
PUNCTURING TABLE

Code Rate	1/2	2/3	3/4	4/5	5/6
Puncturing Pattern	$w^{(0)}:1$ $w^{(1)}:1$	$w^{(0)}:110100$ $w^{(1)}:111111$	$w^{(0)}:010111$ $w^{(1)}:111010$	$w^{(0)}:1011$ $w^{(1)}:0110$	$w^{(0)}:1110001110$ $w^{(1)}:0100110111$

0: punctured; 1: transmitted

message, a VN to CN message, or a channel value. This architecture is based on the flooding scheduling in which the variable node operation in (5) is computed only after all input CN to VN messages are ready. The overall size for message storage is equal to $(2 \cdot q \cdot (J + 1) \cdot (m_s + 2) \cdot I)$ bits, where q is the bit-width of input channel values. If better error-correcting performance is required, larger bit-width q , code memory m_s , and iteration number I are necessary. As a result, the hardware cost and power consumption for message storage are much larger than combinational circuits of CNU and VNUs.

B. LDPC Convolutional Codes for Mobile WiMAX

In this work, we adopt the specification proposed for the IEEE 802.16m standards by Panasonic [25], a rate compatible (491,3,6) LDPC convolutional codes with time period of 3. In order to meet the requirements of next generation mobile communications, IEEE 802.16m is developed to provide higher data rates and lower latency than 802.16e standards. The equations in (6), which are constructed based on the parity check polynomials, define the time-varying convolutional encoder.

$$\begin{aligned}
 &w_t^{(0)}(D)(1 + D^{56} + D^{373}) \\
 &+ w_t^{(1)}(D)(1 + D^{218} + D^{406}) = 0 \\
 &w_{t+1}^{(0)}(D)(1 + D^{197} + D^{457}) \\
 &+ w_{t+1}^{(1)}(D)(1 + D^{22} + D^{491}) = 0 \\
 &w_{t+2}^{(0)}(D)(1 + D^{70} + D^{485}) \\
 &+ w_{t+2}^{(1)}(D)(1 + D^{181} + D^{236}) = 0.
 \end{aligned} \quad (6)$$

$w_t^{(0)}(D)$ and $w_t^{(1)}(D)$ represent the information polynomial and parity polynomial respectively. Due to the time period is only 3, the complexity of hardware implementation is greatly reduced. Besides, this LDPC-CC could support 5 code rates through puncturing. The puncturing patterns are shown in Table I. Fig. 4 is the bit-error-rate (BER) performance of the (491,3,6) time-varying LDPC-CC under AWGN channel. Both the log-BP algorithm and the NMS algorithm are shown in the figure.

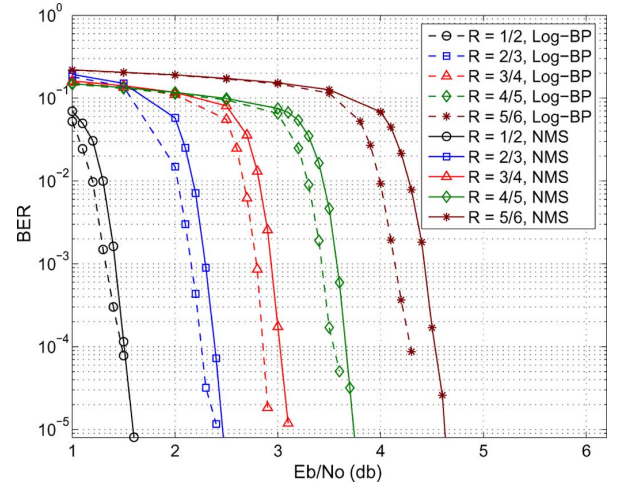


Fig. 4. BER performance under different code-rates with 50 iterations and flooding scheduling.

C. On-Demand Variable Node Activation (OVA) Scheduling

Studies have shown that the message-passing scheduling affects the rate of decoding convergence and computational complexity. If we observe the standard flooding scheduling, the variable node units are activated only once before the values leave the processor. Most messages are shifted while few messages are updated in a processor. Thus, this scheduling is inefficient without utilizing the recently updated information. To increase the convergence speed, sequential scheduling are introduced in decoding LDPC block codes. Recently, an on-demand variable node activation scheduling is proposed in [23], [24] to accelerate the decoding convergence speed for LDPC-CC. The main idea is to change the variable node activation location from the output of the processor to the position right before each check node input. This on-demand variable node activation scheduling is very similar to the layered decoding in LDPC block codes that check nodes could access the most recent messages. We use the same example given in (3) to demonstrate the processor archi-

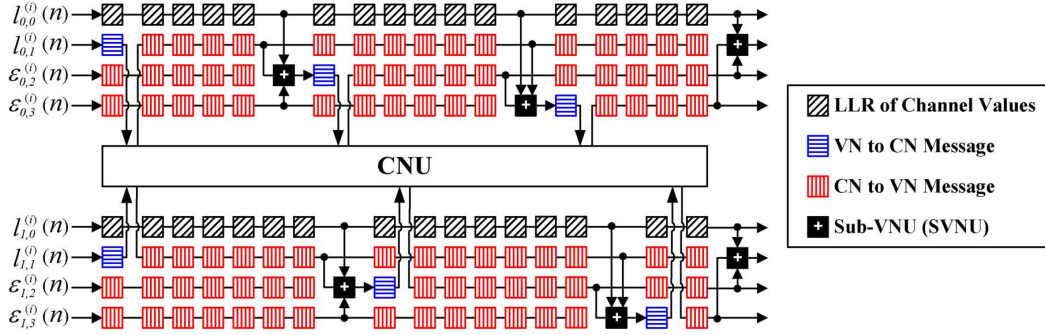


Fig. 5. Processor architecture of on-demand variable node activation scheduling.

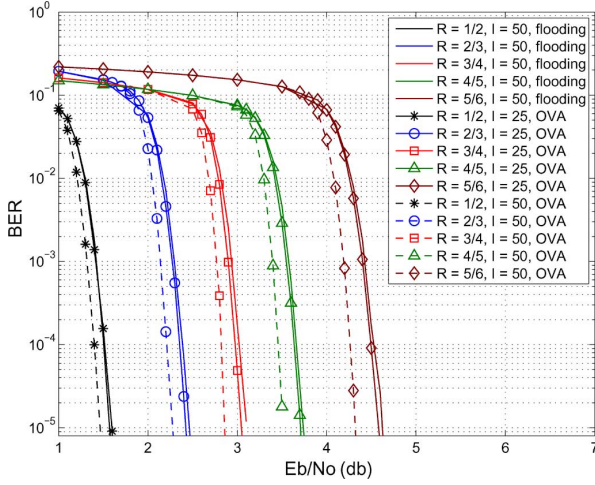


Fig. 6. BER performance of on-demand variable node activation scheduling.

ture of OVA scheduling. Although this example is a time-invariant code, the on-demand scheduling can be applied directly to the case of time-varying codes.

It can be seen from Fig. 5, a VNU can be disassembled into J sub-VNUs (SVNUs) and distributed within a processor. Before each check node unit is activated, the SVNU calculates single VN-to-CN message instead of calculating J VN-to-CN messages in parallel. The major difference from the flooding scheduling is the order of updating procedures, and it is worthy to note that the computational complexity of both scheduling is the same. Fig. 6 shows the floating point BER performance of two scheduling for (491,3,6) LDPC-CC. The performance of OVA with 25 iterations is almost identical to the performance that uses flooding scheduling with 50 iterations. This indicates the OVA scheduling converges twice faster than the standard scheduling due to make use of the most recent information.

III. PROPOSED LDPC-CC DECODER ARCHITECTURE

As explained in the previous section, the OVA scheduling could reduce iteration number by half, but the cost is still large for the code with long constraint length. In order to solve this problem, we propose a modified decoding algorithm which not only avoids the increasing FIFOs but also further reduces the storage of channel values. In the meantime, folding technique suitable for proposed scheduling is also presented to increase throughput; however, it will divide the FIFOs into more pieces

and make it difficult to use memory-based decoder architecture. Hence we will present a hybrid-partitioned FIFO to support large bandwidth requirement of folding technique and also minimize the power consumptions.

A. Proposed Decoding Algorithm Description

The message storage of proposed decoding algorithm is still based on FIFOs. Detail operations of OVA scheduling are shown in Fig. 7(a). Let's observe the first and second VN to CN messages $l_{w,1}^{(i)}(n_1)$ and $l_{w,2}^{(i)}(n_2)$

$$l_{w,1}^{(i)}(n_1) = l_{w,0}^{(i)}(n_1) + \varepsilon_{w,2}^{(i)}(n_1) + \varepsilon_{w,3}^{(i)}(n_1) \quad (7)$$

$$l_{w,2}^{(i)}(n_2) = l_{w,0}^{(i)}(n_2) + \varepsilon_{w,1}^{(i)}(n_2) + \varepsilon_{w,3}^{(i)}(n_2). \quad (8)$$

Since the first row ($j = 0$) stores channel values, the value of $l_{w,0}^{(i)}(n_2)$ is the value of $l_{w,0}^{(i)}(n_1)$ shifted by $n_2 - n_1$ time instances. Furthermore, the fourth row ($j = 3$) stores the third VN to CN message which is unchanged before position n_3 . Therefore $\varepsilon_{w,3}^{(i)}(n_2)$ is also equal to the value of $\varepsilon_{w,3}^{(i)}(n_1)$ shifted by $n_2 - n_1$ time instances. In this case, there are two common terms between (7) and (8). We can obtain $l_{w,2}^{(i)}(n_2)$ from $l_{w,1}^{(i)}(n_1)$ by subtracting the obsolete CN to VN message $\varepsilon_{w,2}^{(i)}(n_1)$ and then adding the immediate CN to VN message $\varepsilon_{w,1}^{(i)}(n_2)$. The modified operation is shown in Fig. 7(b). The computation of modified equation does not need the channel values; consequently the FIFO storing channel values is removed. In fact, the channel values are concealed inside the updated VN to CN messages. Thus the proposed algorithm is named as OVA scheduling with concealing channel values (OVA-CC).

Fig. 8 demonstrates the algorithm-level optimization to accelerate the decoding convergence speed by using the proposed OVA-CC technique. The variable activation location is the same as the original OVA scheduling. The subtraction of the obsolete CN to VN message $\varepsilon_{w,2}^{(i)}(n_1)$ is done by pre-SVNU and the addition of the immediate CN to VN message $\varepsilon_{w,1}^{(i)}(n_2)$ is done by post-SVNU. Comparing with the original OVA scheduling, the combinational logic is similar since the gate count of 3-input adder is near the sum of 2-input adder and 2-input subtractor. However in the proposed scheduling, the channel values ($l_{0,0}^{(i)}(n)$ and $l_{1,0}^{(i)}(n)$) are concealed in VN-to-CN messages and the storage space of channel values can be removed from processors to save memories. In order to conceal the channel values inside VN-to-CN messages, the bit-width

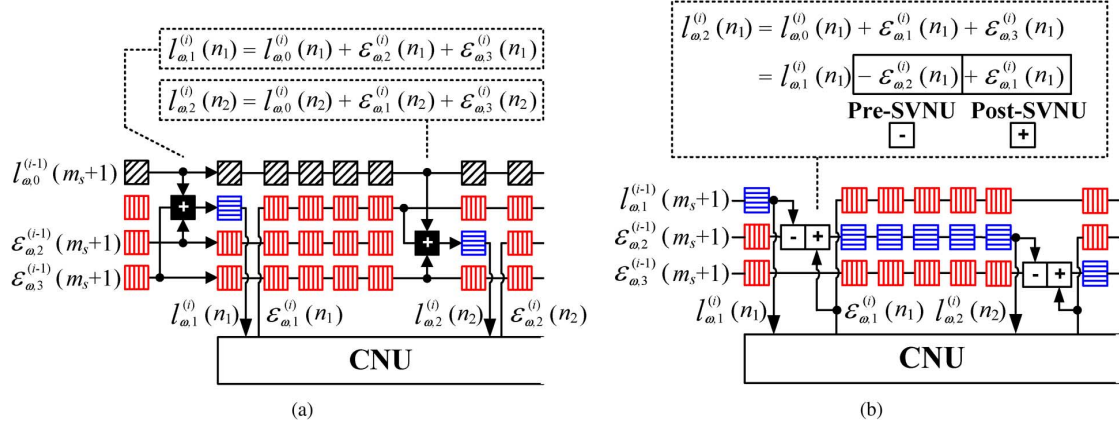


Fig. 7. Modify equations to conceal channel values. (a) Equations of OVA scheduling. (b) Equations of OVA scheduling with concealing channel values.

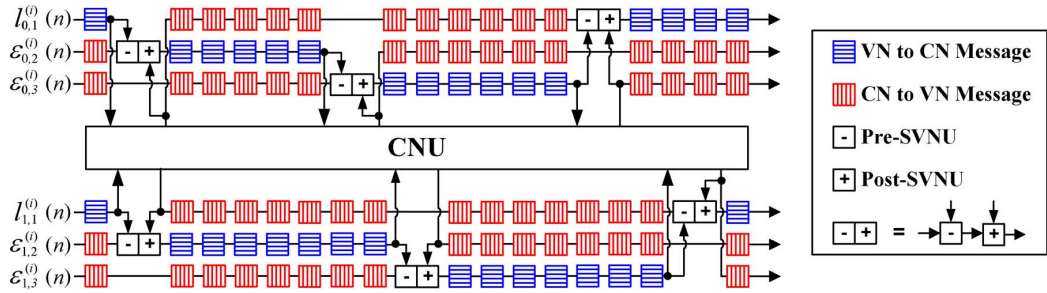


Fig. 8. Processor architecture of on-demand variable node activation scheduling with concealing channel values.

of VN-to-CN messages need to be adjusted to $q + 2$. Therefore the overall FIFO capacity of proposed design is roughly $(2 \cdot (q(J - 1) + (q + 2)) \cdot (m_s + 2) \cdot I)$. The memory saving percentage is $(qJ + 2)/(qJ + q)$.

Fig. 9 shows the fixed point simulation of the (491,3,6) LDPC-CC with five iterations using NMS algorithm with scaling factor 0.875. Comparing to log-BP algorithm with ten processors, the proposed algorithm with five processors can achieve similar or even better performance in all code-rates. Therefore, only half number of processors is required under the same performance, leading to half decoding latency reduction as well.

B. Folding Technique

In the original pipeline decoder architecture, a number of processors are concatenated together to decode on different regions over the Tanner graph simultaneously, thus the decoding is parallel in the iteration dimension. Assume the decoder can operate at f_{clk} MHz clock frequency, since the decoder can decode only one information bit in one cycle, the information throughput will be limited to f_{clk} Mb/s. In order to provide a solution with lower complexity, we propose the folding technique for node level parallelization to design a high throughput LDPC-CC decoder. The idea of folding technique is to look ahead the bits which participate in the decoding operations. Each FIFO line in the conventional processor is folded to ρ FIFO lines, where ρ is defined as the folding factor or the parallelization factor. In other words, each FIFO line is segmented by ρ factor to support required bandwidth. With this modified FIFO structure, sufficient input data could be provided for operation units.

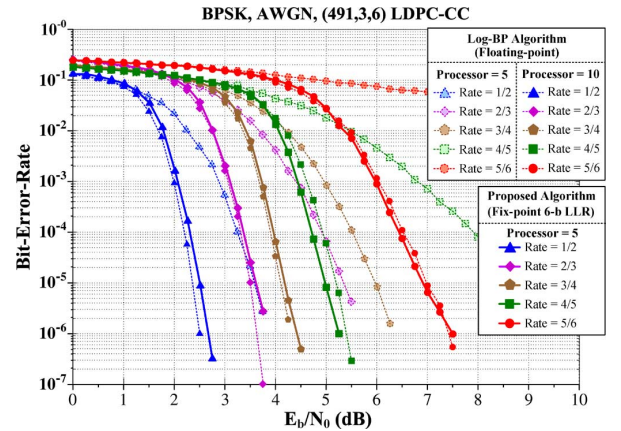


Fig. 9. BER performance of log-BP algorithm (floating-point) and our proposed scheduling in NMS algorithm with scaling factor 0.875 (fixed-point) under AWGN channel.

For instance, Fig. 10 illustrates the architecture of the folding factor $\rho = 3$. For simplicity, only the messages of first codeword symbol are drawn in the figure. In order to handle the computation within ρ time instances, the first step is to add $(\rho - 1)$ pipeline registers before each input as shown in Fig. 10(a). Then each shift register of length $16 + 2$ is folded to 3 rows of shift registers of length 6, as shown in Fig. 10(b). Also, each check node unit and each variable node unit are duplicated to ρ units. The last step is to reconnect the shifter registers such that the operations within $\rho = 3$ time instances are computed in one cycle,

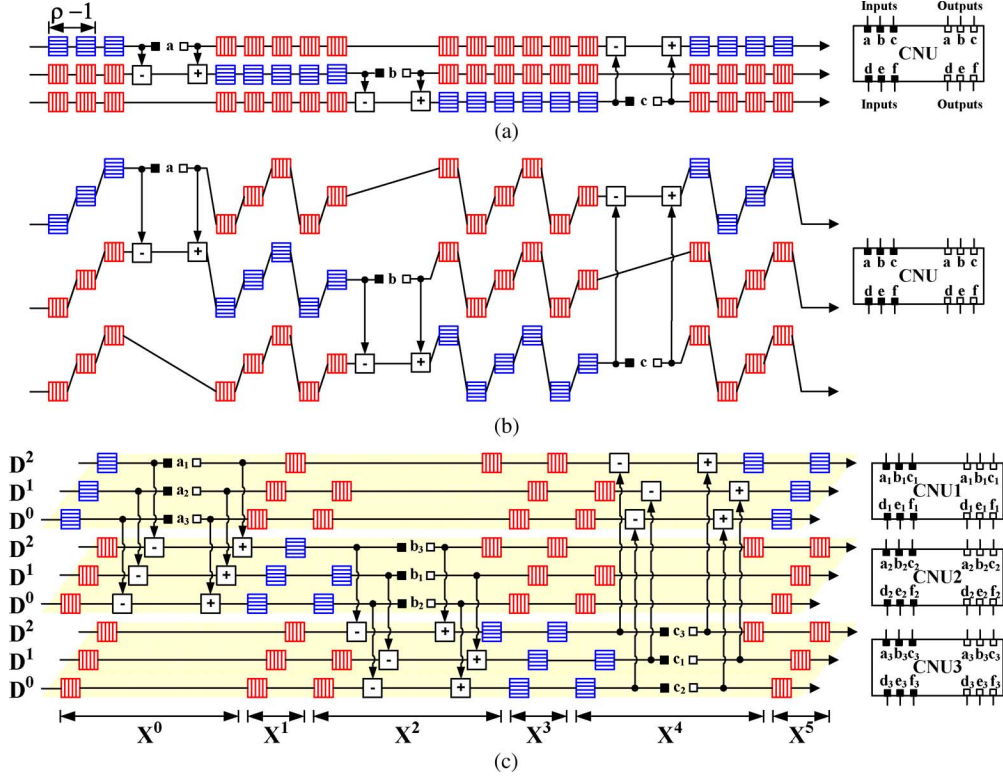


Fig. 10. Node level optimization: folding technique. (ρ is 3 in this example). (a) Add $(\rho - 1)$ pipeline registers before each input. (b) fold each row into ρ rows. (c) duplicate CNU/VSNU and reconnect shift registers.

as shown in Fig. 10(c). Namely, the decoding latency is reduced from 16 cycles to 6 cycles for a unit processor.

Notice that input/output connections of duplicated CNU become complicated after folding. This can be observed in Fig. 10(a) that original input/output (a,b,c) of CNU are regularly located at the first row, second row, and third row. After folding the input/output (a_1, b_1, c_1) of CNU1 are irregularly located at first row, fifth row, and eighth row. The connections will become very complicated when time-varying LDPC-CC is adopted with a large folding factor. In order to solve this problem, we describe the folding architecture by parity-check polynomials. Take the time-invariant LDPC-CC in (3) as an example, the parity-check polynomials with $\rho = 3$ time instances are described by

$$\begin{aligned} D^2 \cdot w_t^{(0)}(D)(1 + D^5 + D^{11}) + D^2 \cdot w_t^{(1)}(D)(1 + D^7 + D^{14}) &= 0 \\ D \cdot w_{t+1}^{(0)}(D)(1 + D^5 + D^{11}) + D \cdot w_{t+1}^{(1)}(D)(1 + D^7 + D^{14}) &= 0 \\ w_{t+2}^{(0)}(D)(1 + D^5 + D^{11}) + w_{t+2}^{(1)}(D)(1 + D^7 + D^{14}) &= 0. \end{aligned}$$

Let $X = D^3$, these equations can be rewritten as

$$\begin{aligned} w_t^{(0)}(D)(D^2 + X^2 \cdot D + X^4 \cdot D) \\ + w_t^{(1)}(D)(D^2 + X^3 + X^5 \cdot D) &= 0 \end{aligned} \quad (9)$$

$$\begin{aligned} w_{t+1}^{(0)}(D)(D + X^2 + X^4) \\ + w_{t+1}^{(1)}(D)(D + X^2 \cdot D^2 + X^5) &= 0 \end{aligned} \quad (10)$$

$$\begin{aligned} w_{t+2}^{(0)}(D)(1 + X \cdot D^2 + X^3 \cdot D^2) \\ + w_{t+2}^{(1)}(D)(1 + X^2 \cdot D + X^4 \cdot D^2) &= 0. \end{aligned} \quad (11)$$

As a result, the polynomials in (9), (10), (11) are used to depict the input/output locations of CNU1, CNU2, and CNU3, respectively. For example, the polynomial $(D^2 + X^2 \cdot D + X^4 \cdot D)$ describes $a_1 = (X^0, D^2)$, $b_1 = (X^2, D^1)$, and $c_1 = (X^4, D^1)$, as shown in Fig. 10(c). Due to the data-dependency, each polynomial with the form of $(D^a \cdot X^\alpha + D^b \cdot X^\beta + D^c \cdot X^\gamma)$ is constrained by $\alpha < \beta < \gamma$ and $a < \rho$, $b < \rho$, $c < \rho$. Consequently, the maximum folding factor of any LDPC-CC occurs when $\alpha = \beta - 1$ or $\beta = \gamma - 1$ is found in any folded polynomials.

Using this approach to (491,3,6) LDPC-CC defined in (6) and letting $X = D^\rho$, we can search the possible folding factors. To simplify the searching procedure, the folding factors are usually the multiples of period for time-varying LDPC-CC. Generally the LDPC-CC with larger constraint length and careful code constructions would allow higher folding factor. After applying folding technique to the time-varying (491,3,6) LDPC-CC with period of 3, the possible folding factors are 3, 6, 9, and 12. The decoder throughput will become $(\rho \times f_{clk})$ Mb/s. Table II lists a comparison of storage requirements and decoding latency for a unit processor under different folding factors for (491,3,6) LDPC-CC. The folding technique primarily duplicates the combinational logic while the sequential circuits are only slightly increased. In other words, the increased percentages of storage bits for $\rho = 3$ and $\rho = 12$ are only 1% and 4.7%, respectively, which is a slight overhead. Furthermore, one basic flip-flop cell occupies about 5.3 times area of minimum NAND2 and the area of CNU and VNU are 690 and 194 times area of minimum NAND2 by synthesized in 90 nm process. Thus, the complexity of one CNU and one VNU are roughly equal to 130-bit and 37-bit registers. The duplication overhead of CNU and VNU

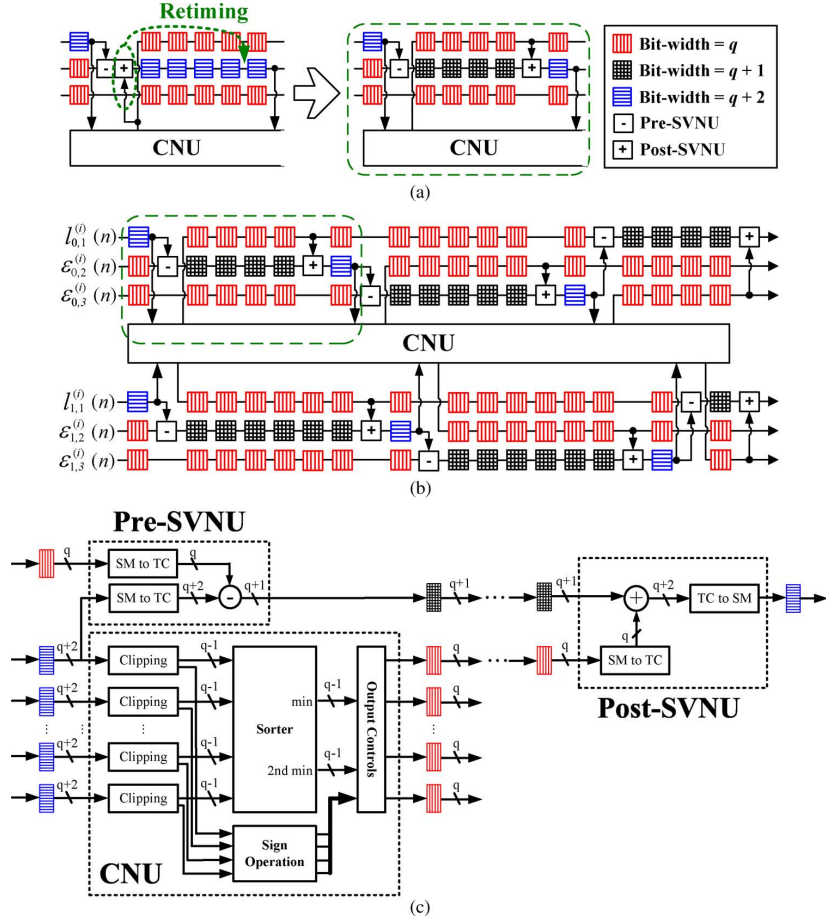


Fig. 11. Retime sub-VNUs to reduce both critical path and bit-width. (a) Retiming of sub-VNUs. (b) Processor architecture with retimed sub-VNUs. (c) Detail circuits of retimed operation units.

TABLE II
HARDWARE COST AND DECODING LATENCY UNDER
DIFFERENT FOLD FACTORS (SINGLE PROCESSOR)

Folding Factor	1	3	12
Required Bits for Storage	23664	23904	24768
Number of CNUs	1	3	12
Number of VNUs	1	3	12
Throughput	f_{clk}	$3 \times f_{clk}$	$12 \times f_{clk}$
Decoding Latency (cycles)	493	166	43

for $\rho = 3$ and $\rho = 12$ are 1.4% and 7.7%, which are still minor as comparing to the overall cost of a processor. It is evident that our approach with 12.4% hardware overhead can increase 12 times throughput, indicating the decoding latency is reduced from 493 cycles to 43 cycles for a single processor. Therefore we choose folding factor 12 for implementation.

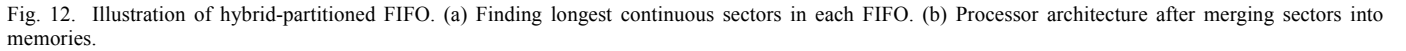
C. Retiming of Sub-VNUs

From previous mentioned decoding scheduling, the channel values are concealed in the CN-to-VN messages. To avoid truncation error, the bit-width of each message should be adjusted. In the situation of q -bit channel values, the summation of one

channel value and two CN-to-VN messages needs $(q + 2)$ -bits. Since the operations of pre-SVNU and post-SVNU are independent, they can be retimed such that the messages between them only need $(q + 1)$ -bits. The procedure of retiming is shown in Fig. 11(a). As long as the computation of sub-VNU is completed before CNU accesses the message, the result is identical to the original operation. In order to achieve a maximum saving in hardware cost, the computation of post-SVNU is moved to the position just before CNU accesses the messages.

Fig. 11(b) depicts the bit-level optimized processor architecture. The critical path of the conventional processor is dominated by the CNU due to large sorters. On the other hand, the critical path of original OVA-CC is CNU plus post-SVNU. With the retiming technique for sub-VNUs, the critical path from CNU to post-SVNU could be diminished by one adder delay. Moreover, the conventional CNU architecture includes conversion from sign-to-magnitude (SM) to two's complement (TC), clipping, sorter, sign operation, output selection, and conversion from TC to SM. To balance the delay time between CNU and SVNU, the conversions of SM to TC and TC to SM could be moved to pre-SVNU and post-SVNU, as shown in Fig. 11(c). As a consequence, the retiming of sub-VNUs could make the critical path smaller than conventional while reducing memory requirements. This technique is especially useful to large constraint length LDPC-CCs for the long distance between two check node inputs.

	6-bit message	7-bit message	8-bit message	Total required bits per processor
Flooding Scheduling	4128	0	0	24768
OVA-CC	2064	0	1032	20640
OVA-CC with Retiming	2064	960	72	19680



For irregular time-varying LDPC-CC with large folding factor, neither register-based FIFO with high power consumption nor memory-based FIFO with serious memory conflict is suitable. In order to make trade-off between bandwidth and power, the hybrid-partitioned FIFO structure is presented to support large bandwidth requirement and also minimize the power consumption. The first step is calculating the length of the longest continuous sectors of every folded row. As shown in Fig. 12(a), the operation in these sectors is simply shifting and power hungry. Then the sectors are merged into one memory bank together, where the depth of the memory bank is the minimum value of the sector lengths. If the original sector is larger than the memory depth, the excess parts are

For the LDPC-CC with larger constraint length, the lengths of continuous sectors within a processor will be longer. Large amounts of data are saved in the memory banks instead of registers, leading to a significant saving in power consumption. Furthermore, if large folding factor is employed, the number of continuous sectors in a processor will increase, and the lengths of continuous sectors will be shortened. These segmented and shortened sectors can still be merged into several unified memory banks. The memory bank is implemented as a circular buffer whose positions for read and write operations are tracked by address pointers. Two-port memories are used in this work

TABLE IV
COMPARISON OF CLOCK TREE LOADING

Folding Factor	12	12	12
Number of Processors	1	1	1
Number of Memories	0	1	3
Clock Buffer	359	278	194
Sink Number	17968	13676	9068
Area (mm ²)	750 × 520	750 × 520	750 × 540

such that read and write operations can be performed in the same clock cycle. The shifting operations in the FIFOs are reduced to achieve a low-power implementation.

In this work, about 50% of messages in each processor are partitioned into three two-port memories. The parameters of these two-port memories are 20 words × 144 bits, 32 words × 76 bits, and 36 words × 144 bits, respectively. The total memory size used in each processor is

$$(36 \times 144) + (20 \times 144) + (32 \times 76) = 10.5 \text{ Kbits.} \quad (12)$$

The power reduction can be observed in the decreased clock tree loading. During the physical design stage, sink is the number of leaf cells connected in the clock tree. The less number the sink is, the less power the clock tree consumes. Table IV lists the clock tree loadings under different FIFO structure. The processor based on hybrid-partitioned FIFO structure occupying area similar to the other two cases can greatly reduce the numbers of clock buffers and sinks. Moreover, merging more continuous sectors into memory banks will eliminate more buffers and sinks. As shown in Table IV the hybrid-partitioned FIFO using 3 memory banks reduces 54% clock buffers as compared to register-based FIFO.

IV. CHIP IMPLEMENTATION

Based on our proposed OVA scheduling with concealed channel values, folding architecture, retimed SVNU, and hybrid-partitioned FIFOs, a test chip of the (491,3,6) LDPC-CC decoder is fabricated in 90 nm 1P9M CMOS process. The chip micrograph is shown in Fig. 13. In this section, we describe the measurement environment, summarize the key characteristics, and compare our ASIC implementation results to other state-of-the-art designs.

A. Measurement Environment and Results

As shown in Fig. 14, the random number generators (RNGs), encoder, additive white Gaussian noise (AWGN) engine, puncture/de-puncture module, and proposed LDPC-CC decoder are implemented in the test chip. The RNG can provide sufficient test patterns for real-time on-chip decoding. Since the decoding latency and decoder throughput are 43×5 cycles and 12-bit per cycle, two identical RNGs are used to avoid a large buffering FIFO (2580-bits) of information sequence. AWGN engine is built up according to the Box-Muller algorithm [26]. Furthermore, the puncture/de-puncture module allows the LDPC-CC

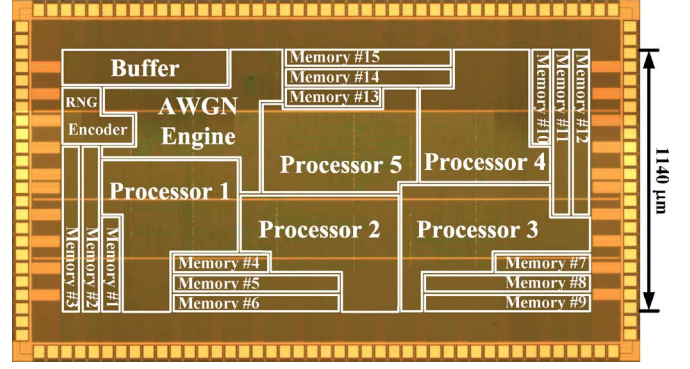


Fig. 13. Chip micrograph.

decoder to support 5 different code-rates from 1/2 to 5/6. The LLR values of punctured bits are set to zeros and then sent to LDPC-CC decoder with five processors. To determine the required bit-width of LLR values, the performance curves of different bit-width choices are simulated under AWGN channel with BPSK modulation. In Fig. 15, the notation of bit-width (6,2) means total 6-bit quantization with 2-bit decimal. Simulation results indicate that both the bit-width (7,3) and (6,2) can achieve less than 0.1 dB implementation loss. Considering the hardware cost, this work implements 6-bit input quantization.

In order to handle chip failures when unexpected errors occurred in any module or any processor, we designed bypass circuits to allow individual testing of each module and each processor. The control signals of all multiplexers can also be configured for following testing modes: BER testing, functional testing, uncoded testing, and variable processors testing. When the control signals of all multiplexers in Fig. 14 are set to ones, the test chip is running at BER testing mode to measure the BER and power under different SNRs. The functional testing mode can verify the correctness of decoder circuit by feeding test vectors from external inputs. Since the input/output pins of the decoder chip are limited, the LLR values from external inputs are serially inputted and stored in the buffer after puncture/de-puncture module. Other configurations of control signals can examine performance of uncoded (bypass and turn off encoder/decoder) and decoding by LDPC-CC decoder with one to five processors. The chip core area including testing circuits is $2.37 \times 1.14 = 2.7 \text{ mm}^2$. The buffer, AWGN engine, and proposed LDPC-CC decoder occupy 5%, 8%, and 83% area, respectively. Therefore the overhead of testing modules is 17%, which is roughly equal to area of one processor.

With the aid of testing circuits, the BER, throughput, and power consumption can be properly measured. Fig. 16(a) shows the measurement results of BER testing and uncoded testing modes. The measured BER curves of five code-rates are identical to Fig. 9. To get net power consumption of proposed decoder, we firstly measure the power of uncoded testing mode, which is 66.7 mW and independent of the SNRs. Then subtracting the uncoded power from total power is the net power consumption of proposed decoder. Our proposed decoder consumes 238 to 285 mW at 198 MHz under 1.2 V supply, while the highest power occurs at code-rate 1/2 and the power of other punctured code-rates are similar. Note that the throughput and

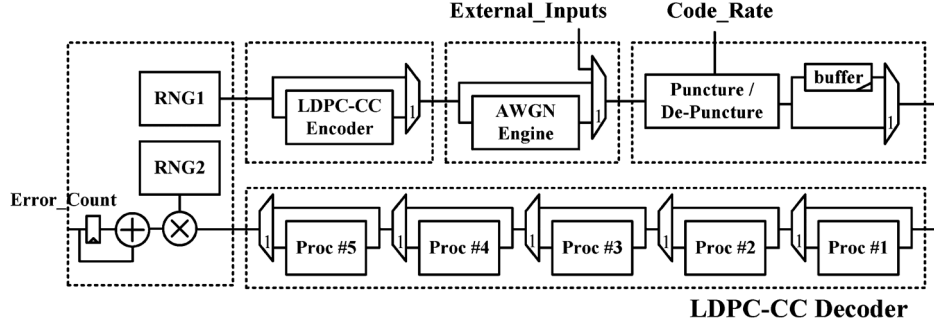


Fig. 14. Testing circuits of proposed decoder.

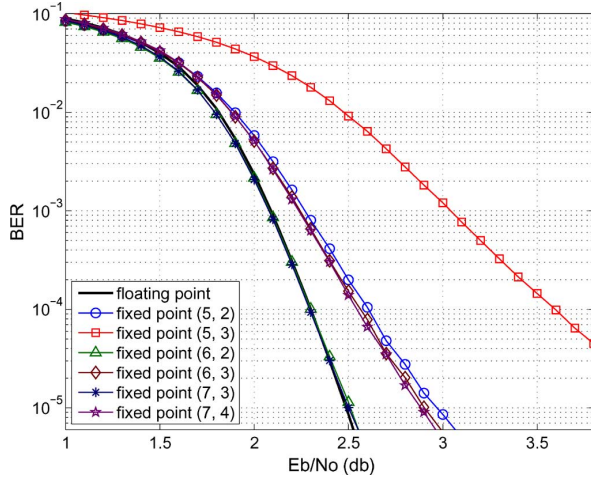


Fig. 15. BER performance of code-rate 1/2 under different bit-widths.

power do not vary a lot in high SNR region because early-termination is not applied.

The measurement result of the test chip at an SNR 2.5 dB under different supply voltages is shown in Fig. 16(b). The information throughput linearly increases as the supply voltage increases. The result shows that the decoder draws 284 mW under 1.2 V supply voltage while running at 198 MHz. Since the folding factor equals 12, the information throughput of the proposed decoder achieves $0.198 \times 12 = 2.37$ Gb/s. When supply voltage is scaled down to 0.8 V, the power is reduced to 90.2 mW with a better energy efficiency of 0.0114 nJ/bit/proc. The Shmoo plot is shown in Fig. 16(c). We choose the SNR 2.5 dB which can achieve a BER of 10^{-5} to simulate the valid range of operating frequency and supply voltage. Table V summarizes key features of the proposed (491,3,6) time-varying LDPC-CC test chip.

B. Comparison With Other LDPC-CC Decoders

A comparison with four state-of-the-art LDPC-CC decoder designs is given in Table VI. Since these designs have different implementation parameters including code memory size m_s , CMOS technology, supply voltage, input quantization, and processor number, it is necessary to apply performance normalization for a fair comparison. Here we assume the growth rate

TABLE V
CHIP SUMMARY

Process	UMC 90nm 1P9M	
Code	(491,3,6) LDPC-CC with T=3	
Code Rate	1/2, 2/3, 3/4, 4/5, 5/6	
Constraint Length	984	
Input Quantization	6 bits	
Chip Utilization	87.8%	
Gate Count (Decoder/Chip)	725K / 867K	
Processor Number	5	1
Memory	52.5Kb	10.5 Kb
Decoder Area	2.240mm ²	0.448 mm ²
Max. Clock Frequency ⁽¹⁾	198 MHz	207 MHz
Max. Data Rate ⁽¹⁾	2.37 Gb/s	2.48 Gb/s
Decoder Power ⁽¹⁾	284mW	-

(1) measured at BER = 10^{-5} without early-termination under 1.2 V supply voltage for code-rate 1/2

of overall complexity for LDPC-CC decoder is linear to code memory m_s . The implementation scaling factors are defined as follows:

$$\begin{aligned}
 &\text{Technology Scaling Factor } (s_t) \\
 &= 90 / (\text{design's technology}(\text{nm})) \\
 &\text{Voltage Scaling Factor } (s_v) \\
 &= 1 / (\text{design's supply voltage}) \\
 &\text{Code Size Scaling Factor } (s_c) \\
 &= 491 / (\text{design's } m_s) \\
 &\text{Bitwidth Scaling Factor } (s_w) \\
 &= 6 / (\text{design's bit - width}) \\
 &\text{Processor Scaling Factor } (s_n) \\
 &= 1 / (\text{design's processor number}),
 \end{aligned}$$

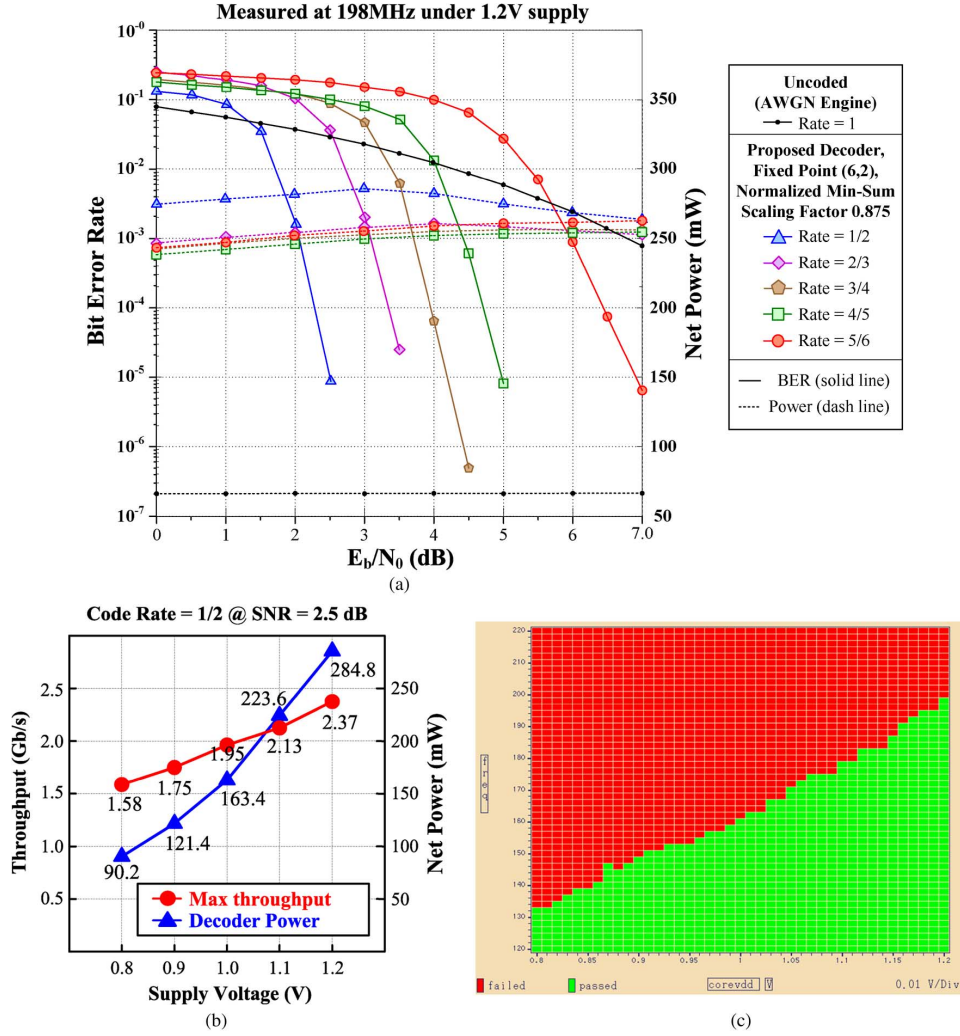


Fig. 16. Measurement Results. (a) Measured BER and power consumption under different SNR conditions. (b) Measured throughput and power consumption under different supply voltages. (c) The Shmoo plot.

where the normalization targets to 90 nm, 1 V supply, $m_s = 491$, 6-bit LLR input, and single processor. Accordingly, the performance scaling factor of each design can be computed by

$$\text{Area Scaling Factor } (s_a) = s_t^2 \times s_c \times s_w \times s_n$$

$$\text{Frequency Scaling Factor } (s_f) = 1/s_t$$

$$\text{Latency Scaling Factor } (s_l) = s_t \times s_c \times s_n$$

$$\text{Power Scaling Factor } (s_p) = 1/s_v^2 \times s_t^3 \times s_c \times s_w \times s_n.$$

The values after normalization are listed with parentheses in Table VI. The results of LDPC-CC decoder in [15] are not normalized because code memory size is near 491 and other implementation parameters are the same to normalization targets. In our design, we directly use measurement results at 1 V instead of normalized values.

As a result, the performance of throughput, latency, power, and hardware cost can be fairly compared in this table. Although the operating frequency of proposed decoder is slowest among five decoders, high throughput can still be achieved by high parallelism. The normalized throughput of proposed decoder is similar to the decoder in [15], but the required area of our design

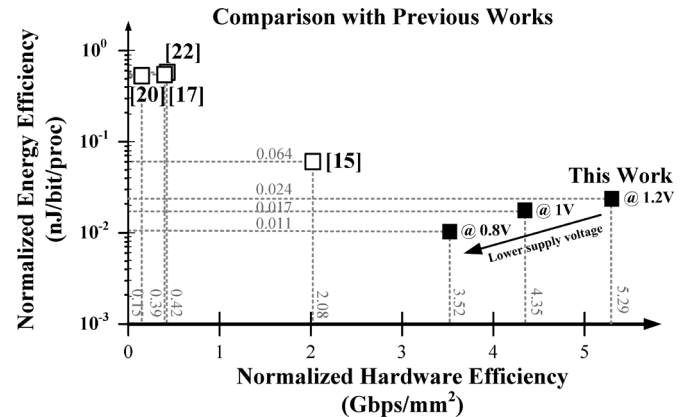


Fig. 17. Comparison of normalized energy efficiency and hardware efficiency among LDPC-CC decoders.

is less than half of the other four decoders. In other words, the proposed decoder architecture successfully raises throughput with 50% reduced complexity. As for the normalized power consumption, our single processor only consumes 32.7 mW, which is only 9%, 4%, and 16% of three decoders in [17], [20],

TABLE VI
COMPARISON WITH STATE-OF-THE-ART OF LDPC-CC DECODERS

	This work	TCAS-I 2010 [15]	TCAS-I 2009 [22]	ISCAS 2008 [20]	JSSC 2007 [17]
LDPC-CC Code	(491,3,6)	(472,3,6)	(128,3,6)	(127,3,5)	(128,3,6)
Code-Rate	1/2, 2/3, 3/4, 4/5, 5/6	1/2	1/2	Programmable	1/2
CMOS Technology (nm)	90	90	90	130	180
Input Quantization (bit)	6	6	8	5	6
Processor Number	5	1	3	1	10
Decoder Area (mm^2)	2.24 (0.448 ^a)	0.924	1.5 (1.44)	5.43 (12.07)	9.9 (0.95)
Max. Frequency (MHz)	198 (163 ^a)	250	600 (600)	200 (289)	175 (350)
Max. Data Rate (Gb/s)	2.37 (1.95 ^a)	2.0	0.6 (0.6)	1.07 (1.55)	0.175 (0.35)
Latency (μs)	1.086 (0.264 ^a)	0.24	0.39 (0.88)	1.2 (3.21)	7.98 (1.53)
Power (mW)	284 (32.7 ^a)	-	368.7 (353.6)	370 (820)	1300 (202)
Energy Efficiency (nJ/bit/proc)	0.024 (0.017 ^a)	0.064	0.2048 (0.589)	0.34 (0.53)	0.76 (0.577)
Hardware Efficiency ^b	(4.35 ^a)	2.08	(0.42)	(0.13)	(0.39)
	Measurement	Synthesis	Measurement	Synthesis	Measurement

(): normalized to $m_s = 491$, 90-nm technology, 1 V supply, 6-bit quantization, and single processor.

a: measured at 1 V supply voltage and normalized to single processor.

b: Throughput/Area (Gbps/ mm^2)

TABLE VII
COMPARISON WITH OTHER FEC DECODERS

	This work	ASSCC 2010 [27]	JSSC 2011 [28]	ISSCC 2010 [29]	DATE 2010 [30]
FEC Type	LDPC-CC	LDPC-BC	LDPC-BC	Turbo Code	Turbo Code
Constraint Length / Block Size	984	672	576 to 2304	3200	All LTE frame sizes
Code-Rate	1/2, 2/3, 3/4, 4/5, 5/6	1/2, 5/8, 3/4, 7/8	1/2, 2/3, 3/4, 5/6	1/3	1/3 to 0.95
CMOS Technology (nm)	90	65	130	130	65
Input Quantization (bit)	6	6	6	5	6
Processor / Iteration	5	5	10	5.5	6.5
Memory (kb)	52.5	0	72.6	129	-
Chip Utilization (%)	87.8	73.3	85	-	-
Decoder Area (mm^2)	2.24	1.4	3.03	3.57	2.1
Max. Frequency (MHz)	198 ^(b)	197 ^(c)	214 ^(d)	302	300
Max. Data Rate (Gb/s)	2.37 ^(b)	3.57 ^(c)	0.73 ^(d)	0.39	0.15
Power (mW)	284 ^(b)	469.7 ^(c)	342 ^(d)	788.9	300
Energy Efficiency (nJ/bit/proc) ^(a)	0.024 ^(b)	0.026 ^(c)	0.047 ^(d)	0.37	0.302
Error-correcting Capability ^(e)	2.5	2.95	2.7	-	-
	Measurement	Measurement	Measurement	Measurement	Post-Layout

^(a) this unit is nJ/bit/iter in LDPC-BC [27], [28] and Turbo code [29], [30]

^(b) measured at BER = 10^{-5} without early-termination under 1.2 V supply voltage for code-rate 1/2

^(c) measured at BER = 10^{-6} without early-termination for code-rate 1/2

^(d) measured without early-termination under 1.2 V supply voltage for code-rate 5/6

^(e) evaluated by required SNR to reach BER = 10^{-5} for code-rate 1/2

[22], respectively. The low power characteristic is mainly benefited by reduction of message storage, relaxation of clock tree buffer/loading, and realization of register-memory-mixed design style. The energy efficiency and hardware efficiency defined as energy per bit per processor and throughput per area are two effective parameters to evaluate different decoders. To emphasize the advantages over other LDPC-CC decoders, the normalized energy efficiency and hardware efficiency are com-

pared in both Table VI and Fig. 17. The proposed decoder can provide more than twice higher throughput per area with much less energy than the other four designs. By sacrificing some throughput, the decoder operated at lower supply voltages can accomplish better energy efficiency if there is low power demand in handheld devices.

The above-mentioned comparison is based on normalization to single processor. If the BER performance is taken into

consideration, the proposed decoder with OVA-CC scheduling converges twice faster than the other four decoders with flooding scheduling. Therefore, this work provides higher throughput, less area, better energy efficiency, higher hardware efficiency, and better error-correcting performance with previously reported LDPC-CC decoders.

C. Comparison With LDPC-BC and Turbo Decoders

The comparison among different FEC decoders is especially difficult since the implementation style and design parameters are distinct from each other. In order to give a rough concept, the performance of two LDPC-BC decoders, two Turbo decoders, and our work are listed in Table VII. Both LDPC-CC and Turbo code can support multiple code-rates easily by puncturing. On the other hand, LDPC-BC can support multiple code-rates only when the parity-check matrix has certain structures. To evaluate the performance, the normalization model used in previous section is applied with the justification of removing the effect of scaling factors s_c and s_n . The normalized areas of [27]–[30] become 2.68, 1.45, 2.05, and 4.03 mm^2 , whereas the throughputs of [27]–[30] are normalized to 2.58, 1.05, 0.56, and 0.11 Gb/s, respectively. Their corresponding energy efficiency are scaled to 0.097, 0.015, 0.146, and 1.368 nJ/bit/iter. The required SNR to reach $\text{BER} = 10^{-5}$ for code-rate 1/2 is also listed to compare the error-correcting capability.

Compared with the LDPC-BC decoder [27], this work has 16% smaller area, superior error-correcting performance, and better energy efficiency with 8% slower throughput. Compared with the LDPC-BC decoder [28], this work has 125% higher throughput and superior error-correcting performance with 54% larger area and 54% larger energy efficiency. Therefore, the proposed LDPC-CC decoder can provide better error-correcting capability than LDPC-BC decoders in [27], [28] and can efficiently trade decoder area for throughput/power, and vice versa. Compared with the turbo decoder [29], this work achieves four times faster throughput and 85% lower energy efficiency with 9% hardware cost. Compared with the turbo decoder [30], this work performs better in aspects of area, throughput, and energy efficiency. Error-correcting performance is not listed for turbo decoder because the code-rates in [29], [30] is 1/3. In conclusion, our proposed LDPC-CC decoder outperforms state-of-the-art designs and has the potential to be one candidate for next-generation communication systems.

V. CONCLUSION

We have presented a LDPC-CC decoder design that targets high-throughput, low-cost and low-power. The test chip of (491, 3, 6) time-varying LDPC-CC supporting five code-rates is implemented in 90 nm CMOS technology. The decoder containing 5 processors occupies 2.24 mm^2 and provides twice faster decoding convergence speed. Maximum throughput 2.37 Gb/s is measured under 1.2 V supply with 0.024 nJ/bit/proc energy efficiency. The power can be scaled down to 90.2 mW with lower throughput 1.58 Gb/s at 0.8 V supply. The proposed design methodologies would make LDPC convolutional codes more competitive to the other error-control codes.

ACKNOWLEDGMENT

The authors would like to thank UMC for fabrication of the test chip, Chip Implementation Center for providing the CAD tools and measurement equipment, and ATU program—eNES.

REFERENCES

- [1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
- [2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 33, no. 6, pp. 457–458, Mar. 1997.
- [3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Feb. 1999.
- [4] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.
- [5] J. Zhang and M. Fossorier, "Shuffled belief propagation decoding," in *Proc. 36th Asilomar Conf. Signals, Syst., Comput.* 2002, Pacific Grove, CA, Nov. 2002, vol. 1, pp. 8–15.
- [6] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.
- [7] *IEEE 802.11 Wireless LANs WWiSE Proposal: High Throughput Extension to the 802.11 Standard*, IEEE Std. 11-04-0886-00-000n.
- [8] *Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)*, IEEE Std. P802.15.3c-DF8, 2009.
- [9] *Part 16: air interface for fixed and mobile broadband wireless access systems*, IEEE Std. P802.16e-2005, Oct. 2005.
- [10] Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications ETSI Std. EN 302 307, Rev. 1.1.2, 2006.
- [11] A. J. Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181–2191, Jun. 1999.
- [12] E. Matus, M. Tavares, M. Bimberg, and G. Fettweis, "Towards a GBit/s programmable decoder for LDPC convolutional codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, New Orleans, LA, May 2007, pp. 1657–1660.
- [13] M. Bimberg, M. Tavares, E. Matus, and G. Fettweis, "A high-throughput programmable decoder for LDPC convolutional codes," in *Proc. IEEE Int. Conf. Appl.-Specific Syst., Arch., Process. (ASAP)*, Montreal, QC, Canada, Jul. 2007, pp. 239–246.
- [14] Z. Chen, S. Bates, and W. Krzymien, "High throughput parallel decoder design for LDPC convolutional codes," in *Proc. IEEE Int. Conf. Circuits Syst. Commun. (ICCSC)*, Shanghai, China, May 2008, pp. 35–39.
- [15] Z. Chen, T. Brandon, D. Elliott, S. Bates, W. Krzymien, and B. Cockburn, "Jointly designed architecture-aware LDPC convolutional codes and high-throughput parallel encoders/decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 836–849, Apr. 2010.
- [16] R. Swamy, S. Bates, and T. Brandon, "Architectures for ASIC implementations of low-density parity-check convolutional encoders and decoders," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Kobe, Japan, May 2005, vol. 5, pp. 4513–4516.
- [17] R. Swamy, S. Bates, T. Brandon, B. Cockburn, D. Elliott, J. Koob, and Z. Chen, "Design and test of a 175-Mb/s, rate-1/2 (128,3,6) low-density parity-check convolutional code encoder and decoder," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2245–2256, Oct. 2007.
- [18] S. Bates and G. Block, "A memory-based architecture for fpga implementations of low-density parity-check convolutional decoders," in *IEEE Intern. Symp. Circuits Syst. (ISCAS)*, Kobe, Japan, May 2005, vol. 1, p. 336.
- [19] S. Bates, Z. Chen, L. Gunthorpe, A. Pusane, K. Zigangirov, and D. Costello, "A low-cost serial decoder architecture for low-density parity-check convolutional codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 7, pp. 1967–1976, Aug. 2008.
- [20] M. Tavares, E. Matus, S. Kunze, and G. Fettweis, "A dual-core programmable decoder for LDPC convolutional codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Seattle, WA, May 2008, pp. 532–535.

- [21] M. Tavares, E. Matus, and G. Fettweis, "On the structured parallelism of decoders for LDPC convolutional codes - an algebraic description," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Taipei, Taiwan, May 2009, pp. 2453–2456.
- [22] T. Brandon, J. Koob, L. van den Berg, Z. Chen, A. Alimohammad, R. Swamy, J. Klaus, S. Bates, V. Gaudet, B. Cockburn, and D. Elliott, "A compact 1.1-Gb/s encoder and a memory-based 600-Mb/s decoder for LDPC convolutional codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 5, no. 5, pp. 1017–1029, May 2009.
- [23] A. Pusane, M. Lentmaier, K. Zigangirov, and D. Costello, "Reduced complexity decoding strategies for LDPC convolutional codes," in *Proc. Int. Symp. Inform. Theory (ISIT)*, Chicago, IL, 2004, p. 490.
- [24] A. Pusane, A. Feltstrom, A. Sridharan, M. Lentmaier, K. Zigangirov, and D. Costello, "Implementation aspects of LDPC convolutional codes," *IEEE Trans. Commun.*, vol. 56, no. 7, pp. 1060–1069, Jul. 2008.
- [25] *Amendment Text Proposal on Rate Compatible LDPC-Convolutional Codes*, IEEE Std. C802.16m-09/0339, 2009.
- [26] B. Alimohammad, S. Fard, and C. Schlegel, "A compact and accurate Gaussian variate generator," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 5, pp. 517–527, May 2008.
- [27] S.-Y. Hung, S.-W. Yen, C.-L. Chen, H.-C. Chang, S.-J. Jou, and C.-Y. Lee, "A 5.7Gbps row-based layered scheduling LDPC decoder for IEEE 802.15.3c applications," in *Proc. IEEE Asian Solid-State Circuits Conf. (ASSCC)*, Beijing, China, Nov. 2010, pp. 1–4.
- [28] B. Xiang, D. Bao, S. Huang, and X. Zeng, "An 847-955Mb/s 342-397mw dual-path fully-overlapped QC-LDPC decoder for WiMAX system in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1416–1432, June 2011.
- [29] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "A 390Mb/s 3.57 mm² 3GPP-LTE turbo decoder ASIC in 0.13 μm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf. erence Dig. Tech. Papers (ISSCC)*, San Francisco, CA, 2010.
- [30] M. May, T. Ilseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE turbo code decoder," in *Proc. Design, Autom., Test Eur. Conf. Exhibition (DATE)*, Dresden, Germany, Mar. 2010, pp. 1420–1425.



Chih-Lung Chen received the B.E. and M.Sc. degrees in electronics engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 2004 and 2006, respectively. He is currently pursuing the Ph.D. degree in electronics engineering at NCTU.

His general research interests include very large-scale integration (VLSI) implementation of error control codes and wireless communication systems.



Yu-Hsiang Lin received the B.E. and M.Sc. degrees in electronics engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 2009 and 2011, respectively.

His research interests include error control coding and very large-scale integration (VLSI) architectures.



Hsie-Chia Chang received the B.Sc., M.Sc., and Ph.D. degrees in electronics engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1995, 1997, and 2002, respectively.

From 2002 to 2003, he was with OSP/DE1 in MediaTek Corp., working in the area of decoding architectures for Combo single chip. In 2003, he joined the faculty of the Electronics Engineering Department, NCTU, where he is currently a Professor. His research interests include algorithms and VLSI architectures in signal processing, especially

for error control codes and crypto-systems. Recently, he also commits himself to design high code-rate ECC schemes for flash memory and multi-Gb/s chip implementations for wireless communications.



Chen-Yi Lee (M'01) received the B.Sc. degree from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1982, and the M.Sc. and Ph.D. degrees from Katholieke University Leuven (KUL), Leuven, Belgium, in 1986 and 1990, respectively, all in electrical engineering.

From 1986 to 1990, he was with IMEC/VSDM, working in the area of architecture synthesis for DSP. In February 1991, he joined the faculty of the Electronics Engineering Department, NCTU, Hsinchu, Taiwan, where he is currently a Professor and Dean of Research and Development Office. His research interests mainly include VLSI algorithms and architectures for high-throughput DSP applications. He is also active in various aspects of high-speed networking, system-on-chip design technology, very low power designs, and multimedia signal processing. In these areas, he has published more than 180 papers and holds decades of patents. He served as the Director of Chip Implementation Center (CIC), an organization for IC design promotion in Taiwan (2000/8–2003/12), and the microelectronics program coordinator of Engineering Division under National Science Council of Taiwan (2003/1–2005/12). He was the former IEEE CAS Taipei Chapter Chair.