An Efficient Multirate LDPC-CC Decoder With a Layered Decoding Algorithm for the IEEE 1901 Standard

Yun Chen, Qichen Zhang, Di Wu, Changsheng Zhou, and Xiaoyang Zeng

***Abstract*—An area-efficient multirate low-density parity-check convolutional code (LDPC-CC) decoder is presented in this brief. Using the layered decoding algorithm, the decoder achieves a better performance than the message-passing algorithm; the extrinsic-message storing is switched from variable node based to check node based. Then, using the normalized min-sum (NMS) algorithm, the extrinsic messages can be reduced to the first and second minimum absolute values, the position index of the first minimum absolute value, the signs of all extrinsic messages, and the product of all the signs. A memory-based application-specific integrated circuit architecture of the LDPC-CC decoder that supports these methods is proposed for the IEEE 1901 standard. Based on a SMIC 130-nm complementary metal–oxide–semiconductor process, a decoder that can support all the code rates of the LDPC-CCs defined in IEEE 1901 (1/2, 2/3, 3/4, 4/5) is fabricated and evaluated. The proposed decoder attains a maximum throughput of 300 Mb/s at a maximum operating frequency of 180 MHz. The core area is 3.55 mm^2 with ten processors. The average power consumption is 200.4 mW at code rate 4/5 and a frequency of 180 MHz, and the power efficiency is 66.8 pJ/bit/proc. The very large scale integration results show that the decoder is both memory and area efficient.**

***Index Terms*—IEEE 1901, layered decoding algorithm, low-density parity-check convolutional code (LDPC-CC).**

I. Introduction

LOW-DENSITY parity-check (LDPC) code, introduced by Gallager [1], is widely used in digital broadcast and communication systems for its remarkable performance that can approach the Shannon limit.

In this brief, we have integrated the layered decoding algorithm, which was introduced in [3] for LDPC-convolutional code (CC) [2]. Then, based on the layered decoding algorithm and the NMS algorithm [4], [5], a new architecture is proposed to support the multirate LDPC-CC codes in the IEEE 1901 standard. To verify the proposed architecture, a memory-based decoder, which can support all the code rates of LDPC-CCs in the IEEE 1901 standard, is evaluated. The layered decoding algorithm can achieve a faster convergence speed than the message-passing (MP) algorithm. To the best of our knowledge, this work is the multirate LDPC-CC decoder without puncturing, and the first one can support the IEEE 1901 communication standard.

The rest of this brief is organized as follows. Section II first briefly introduces the LDPC-CC codes. In Section III we discuss the layer decoding algorithm. The simulation results are also shown. Section IV presents the very-large-scale integration (VLSI) implementation of the decoder, and the VLSI results follow in Section V. Section VI concludes this brief.

Manuscript received July 25, 2014; accepted September 15, 2014. Date of publication October 14, 2014; date of current version December 1, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61306043, by the Shanghai Natural Science Funding under Grant 13ZR1401800, by the National Science and Technology Major Project of China under Grant 2011ZX03003-003-0of 3, and by the Open Project of the State Kay Laboratory of ASIC & System under Grant 12KF006. This brief was recommended by Associate Editor Z. Zhang.

The authors are with Fudan University, Shanghai 200433, China (e-mail: chenyun@fudan.edu.cn; xyzeng@fudan.edu.cn).

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

Digital Object Identifier 10.1109/TCSII.2014.2362721

II. Introduction of LDPC-CC

LDPC-CC is defined by a periodically time-varying and infinite parity-check matrix H with memory of M. However, the transposed form H^T is generally used. A rate $R = b/c$ LDPC-CC can be defined as

$$H^T = \begin{pmatrix} H_0^{(0)} & H_1^{(1)} & \cdots & H_t^{(M)} & & \\ & H_1^{(0)} & \cdots & H_t^{(M-1)} & H_{t+1}^{(M)} & \\ & & \ddots & \vdots & \vdots & \ddots \end{pmatrix} \quad (1)$$

where $H_t^{(m)}$, $m = 0, 1, \ldots, \mathrm{M}$, $t = 0, 1, \ldots$, are $c \times (c-b)$ periodically time-varying submatrices. $H_t^{(0)}$ must have full rank and $H_t^{(m)} = H_{t+T_p}^{(m)}$ (T_p is the period of the code) are satisfied for all t and m values.

The LDPC-CCs used in IEEE 1901 are also defined by using check polynomials. All the bits can be divided into two types: the systematic bits and the parity bit. The systematic bits may be divided into several types. For example, a rate 4/5 code contains four different systematic bits: X0, X1, X2, and X3. The parity bit can be expressed as P.

III. Layered Decoding Algorithm

There are some nomenclatures, and corresponding notations will be defined: I_v is the channel intrinsic logarithm likelihood ratio of variable node v. L_v is the posterior message of variable node v. $Z_{c\to v}$ is the extrinsic message from check node c to variable node v. $S_{v\to c}$ is the prior message from variable node v to check node c. α is the normalized factor. $N_{(c)}$ is the set of variable nodes that have a connection with check node c. $M_{(v)}$ is the set of check nodes that have a connection with variable node v. "\" is the symbol for exclusion. $\mathrm{sign}(x)$ is the symbol for computing the sign of x, and x_v is the hard decision value of variable node v.

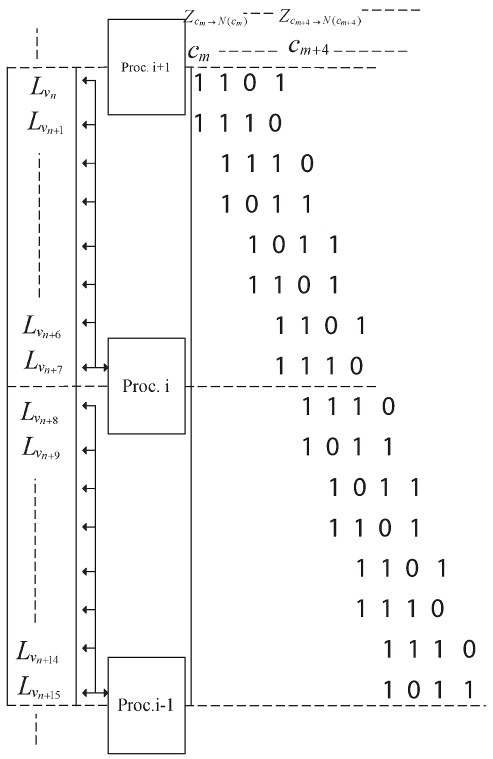

Fig. 1. Section of the FIFO of a decoder for a (3,3,6) LDPC-CC where the layered decoding algorithm is adopted.

A. Description of the Layered Decoding Algorithm

Compared with the min-sum algorithm, the NMS algorithm can increase the decoding performance significantly. Based on NMS, the layered decoding algorithm can be described as follows.

Step 1: Initialization. The posterior messages are filled with ∞, and the extrinsic messages are filled with zeros.

Step 2: Shifting step. The newly received intrinsic message is shifted into all first-in first-out (FIFO) queues and fills the intrinsic message and three prior messages of the first variable node of Processor 0 (Proc. 0). The previous messages for all the variable nodes are shifted one by one. If the newly received nodes are parity bits, then go to Step 3; otherwise, repeat Step 2.

Step 3: Vertical operation. Processors update the corresponding extrinsic messages. Take Fig. 1 for example, Proc. i-1 will first update the prior messages $S_{v_k\to c_{m+7}}$, as shown in (2), and then, the extrinsic messages $Z^{\text{new}}_{c_{m+7}\to v_k}$ will be immediately updated, as shown in (3). Here, $v_k \in N(c_{m+7})$. Then, the updated $Z^{\text{new}}_{c_{m+7}\to v_k}$ will overwrite $Z^{\text{old}}_{c_{m+7}\to v_k}$ in the storage units. Thus

$$S_{v_k\to c_{m+7}} = L^{\text{old}}_{v_k} - Z^{\text{old}}_{c_{m+7}\to v_k} \tag{2}$$

$$Z^{\text{new}}_{c_{m+7}\to v_k} = \alpha \times \prod_{v_i\in N(c_m+7)\backslash v_k} \text{sign}\left(S_{v_i\to c_{m+7}}\right) \times \min_{v_i\in N(c_m+7)\backslash v_k} \left|S_{v_i\to c_{m+7}}\right|. \tag{3}$$

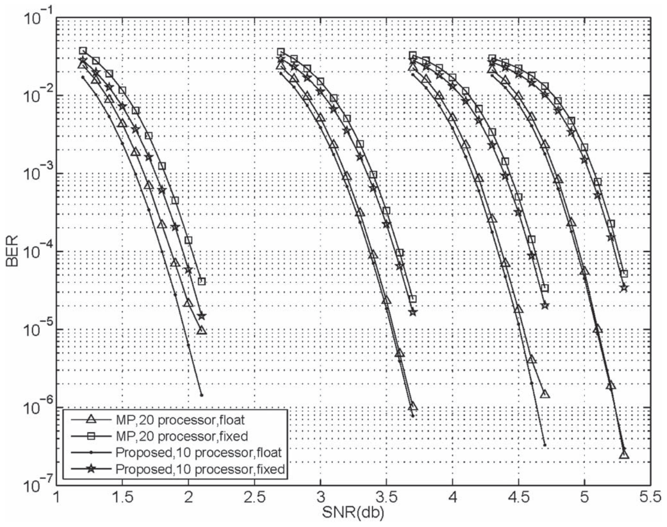

Fig. 2. BER curves for all code rates (1/2, 2/3, 3/4, 4/5) of the LDPC-CC used in the IEEE 1901 standard.

Step 4: Horizontal operation. Processors update the corresponding posterior messages immediately after the prior messages, and the extrinsic messages are updated. Take Fig. 1, for example; Proc. i-1 will update $L^{\text{new}}_{v_k}$, as shown in (4). Here, $v_k \in N(c_{m+7})$. Then, the updated $L^{\text{new}}_{v_k}$ will overwrite $L^{\text{old}}_{v_k}$ in the storage units. Thus

$$L^{\text{new}}_{v_k} = S_{v_k\to c_{m+7}} + Z^{\text{new}}_{c_{m+7}\to v_k}. \tag{4}$$

In the layered decoding algorithm, the storage of the extrinsic messages can be check node based, and extrinsic messages from the same check node can be stored together. Then, based on NMS, the extrinsic messages for each check node can be reduced to the first and second minimum absolute values, the position index of the first minimum absolute value, the signs of all extrinsic messages, and the product of all the signs.

B. Performance of the Layered Decoding Algorithm

The bit error rate (BER) curves for all code rates defined in IEEE 1901 are shown in Fig. 2. All the simulations are carried out based on an additive white Gaussian noise channel with binary phase-shift keying modulation, and the normalized factor is 0.75 in both float and fixed point mode. In fixed point mode, posterior messages and prior messages are quantized to 8 bits with a sign bit, a 4-bit integer, and a 3-bit fraction, whereas extrinsic messages are quantized to 6 bits with a sign bit, a 2-bit integer, and a 3-bit fraction.

The BER curves for all code rates defined in IEEE 1901 are shown in Fig. 2. The performances of on-demand variable node activation (OVA) and the layered decoding algorithm are almost identical, and it is impossible to distinguish them if they are drawn on the same figure; hence, the performance curves for the algorithm OVA are not indicated in this figure. From the BER curves, we can see that the layered decoding algorithm is more efficient and the performance using only half the number of processors is much better than the original MP algorithm.

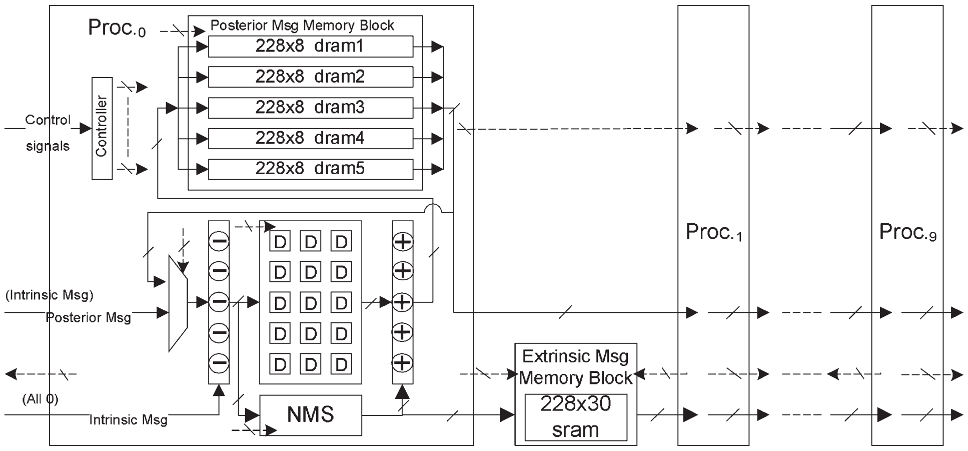

Fig. 3. Architecture of the proposed LDPC-CC decode.

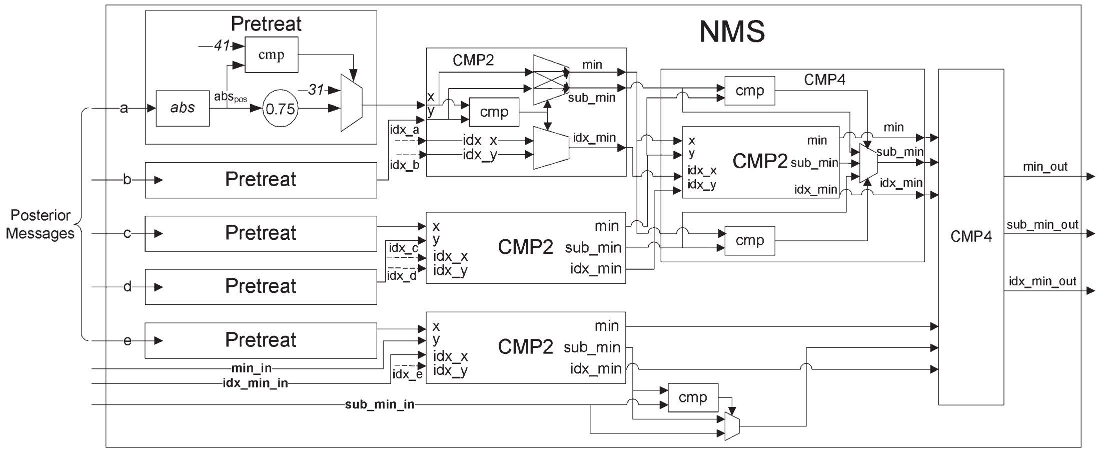

Fig. 4. Main architecture of the NMS block and the extrinsic-message storage bits.

IV. VLSI Implementation

Here, we will introduce the proposed memory-based architecture.

A. Proposed Memory-Based Architecture for the Layered Decoding Algorithm

We propose a memory-based architecture as shown in Fig. 3. The decoder contains ten processors, and every processor has the same architecture as Proc. 0. The extrinsic messages that are updated via Proc. i will never be used again in Proc. i; hence, the updated extrinsic messages can be passed directly to the next processor. Then, in the proposed architecture, the extrinsic messages are stored outside the processor, and only nine extrinsic-message storage blocks are needed because the updated extrinsic messages from the last processor are useless and do not need to be stored.

As Fig. 3 shows, each processor has five parts: 1) a local controller that generates all the control signals; 2) a substractor block that updates the prior messages from the old posterior messages and extrinsic messages; 3) an NMS block that updates the extrinsic messages from the prior message; 4) an adder block that updates the posterior messages from the updated prior messages and extrinsic messages; and 5) a posterior message memory block that stores the posterior messages. This block is made up of five dual-port memories with a width of 8 and a depth of 228.

There are five types of code bits (X0, X1, X2, X3, and P), and the different code bits have different delay factors in the check polynomials. Consequently, it will be much more convenient for the decoder to store the different kinds of code bits in different memories. Five posterior message memories have been adopted in each processor in the proposed decoder. As each kind of code bit has three delay factors in the check polynomials, each memory must finish three reading and three writing operations when processing a single check node. For this reason, dual-port memory is chosen, and three clock cycles are required in order to handle these operations. As a result, there are five dual-port memories, five substractors, and five adders in the corresponding blocks of each processor. Some of the hardware can be disabled to reduce power consumption when decoding rate 1/2, 2/3, and 3/4 codes.

B. NMS Block and Extrinsic-Message Storage Architecture

The main architecture of the NMS block is shown in Fig. 4. The operation of signs of the extrinsic messages is not indicated, because we can simply use the signs for the corresponding posterior messages as the signs for the extrinsic messages. As previously described, the maximum degree of parallelism

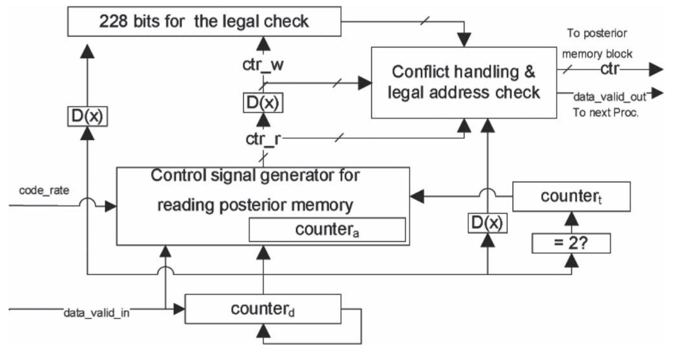
228 bits for the legal check
ctr_w
D(x)
D(x)
Conflict handling & legal address check
To posterior memory block
ctr
data_valid_out
To next Proc.
ctr_r
code_rate
Control signal generator for reading posterior memory
countera
countert
D(x)
= 2?
data_valid_in
counterd

Fig. 5. Main architecture of the controller in each processor.

for message operations in the proposed decoder is five; hence, there are five posterior messages at the input port of the NMS block. The min_in/sub_min_in and idx_in would both be fed with 31/0, where 31 is the maximum value of unsigned 5-bit data, and the index will be simply set to 0, or the corresponding signals of the output, depending on whether the NMS operation is being performed for the first time in a layer. Hence, there are seven input messages for the NMS block.

The absolute value abs_{pos} of the posterior messages is calculated first; then, this value is multiplied by 0.75 (implemented using $abs_{\text{pos}} - 0.25 \times abs_{\text{pos}}$) to complete the normalize operation. The posterior message in the proposed decoder is 8 bits, whereas the extrinsic message is 6 bits; hence, we need to check whether there is overflow in the posterior messages. We compare abs_{pos} with 41, as values bigger than 41 will lead to overflow $(42 \times 0.75 = 31.5)$, and will select the correct extrinsic message based on the result of the comparisons. The maximum value for min/sub_min would be 31.

After the correct extrinsic messages are generated, many comparisons will be required before we can obtain the min/sub_min and idx_min signals. To speed up this operation, three comparison steps can be used to complete the NMS operation: Step 1) Comparisons between original extrinsic messages, which helps determine the relationship between two extrinsic messages. After this step, only six out of seven input messages are left; Step 2) The min/sub_min and idx_min of the previous two sets of min/sub_min and idx_min signals are calculated using a $CMP4$ block. Subsequently, four messages out of the six messages from step 1 are left; Step 3) Another $CMP4$ block calculates the final result from the four messages from step 2.

C. Main Architecture for Controller in Each Processor

The main architecture for the controller in each processor is shown in Fig. 5. $D(x)$ denotes the delay units. The $data\_valid$ signal indicates that new posterior messages are shifted in by the previous processor. The $code\_rate$ signal defines which code rate is used for the current data frame.

The $counter_d$ module is the counter for selecting the delay factor. There are three delay factors for each kind of code bit in the check polynomials. Hence, the value of this counter will only be 0, 1, or 2. The initial value is 0 and will not change to another number until the $data\_vaild$ signal arrives. When the $data\_vaild$ signal arrives, the *Proc.* can finish the decoding of the corresponding check polynomials, as it can read other

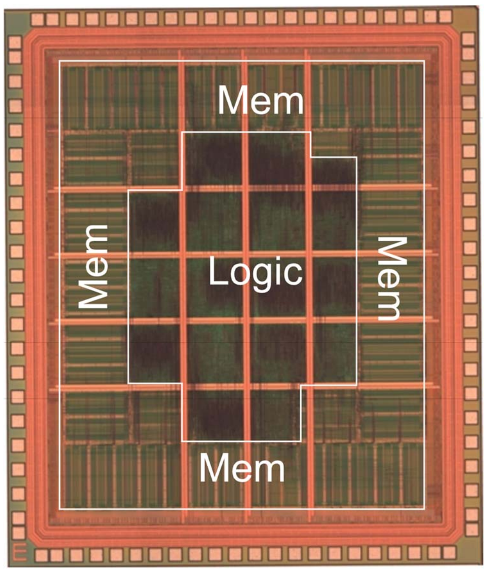

Fig. 6. Image of the die for the proposed LDPC-CC decoder.

posterior messages from the posterior message memory block within itself. Hence, this counter will keep counting until its value returns to 0. The $counter_t$ module is the counter for selecting the check polynomials. The value of this counter will only be 0, 1, or 2 as well, as the period of the LDPC-CCs used in the IEEE 1901 standard is 3. This counter will only toggle when $counter_d$ is 2, which indicates that the information for the current check polynomial has been fully used. The $counter_a$ module is the counter for the address and will count from 0 to 227, as the depth of the posterior memory is 228.

V. Results

The proposed decoder was fabricated using SMIC 0.13 μm 1.2 V 8—metal standard complementary metal–oxide–semiconductor technology, and an image of the die is shown in Fig. 6. The decoder attains a maximum throughput of 300 Mb/s at 180 MHz. (The system's demand is defined as 220 Mb/s in the IEEE 1901 standard.) The core area is 1.7 mm $\times$ 2.09 mm with ten processors. The average power consumption increases as the code rates increase and is 138 mW at rate 1/2, 162 mW at rate 2/3, 188.4 mW at rate 3/4, and 200.4 mW at rate 4/5 when operating at 180 MHz. The energy efficiency (pJ/bit/proc.) is 115, 90, 78.5, and 66.8 at code rate, respectively. The energy efficiency is lower at a higher code rate because the increase in throughput is greater than the increase in power consumption for higher code rates.

Table I shows a comparison of the specifications and performance for the proposed decoder with other LDPC-CC decoders. The area scaling factors used are 2 and 0.5 for 90- and 180-nm technology, respectively. The energy scaling factors used are $2(0.5) \times (target\ voltage/1.2) \times (target\ input\ quantization/8)$.

From Table I, it can be seen that the proposed decoder memory efficiency is better than that of other decoders, because only one posterior message per variable node and one set of extrinsic messages are needed to be stored by the proposed decoder. In addition, because of the better memory efficiency, which can greatly reduce the hardware resources required for storage, the area efficiency of the proposed decoder is better than that

TABLE I
COMPARISON WITH OTHER LDPC-CC DECODERS

	Proposed	[6]	[7]	[8]	[9]	[10]
Code Rate	1/2,2/3,3/4,4/5	1/2,2/3,3/4,4/5,5/6(1)	1/2	1/2	1/2	1/2
Memory(Kb)	152.76	52.5(1)	221.28	67	-	23.04
Frequency(MHz)	180	175	256.4	198	600	250
Core Area(mm^2 scale to 130nm)	3.55	4.48	10.72	4.45	3	1.848
Memory Efficiency (Bit/Constraint length/proc.)	13.46	21.34	20.95	25.97	-	23.04
Area Efficiency (um^2/proc./Constraint length, scaled to 130nm)	295.2	910.6	1015	1919	3876	1925
Throughput (Gbps)	0.3	2.37	1.025	0.175	0.6	2.0
Power (mW)	138, 162, 188.4, 200.4	284.8	682	1300	368.7	-
Energy Efficiency (pj/bit/proc., Scaled to 130 nm and 8 bit input quantization)	115, 90, 78.5, 66.8	64	161.3	337.7	491.52	-
Technology (nm, V)	130,1.2	90,1.2	90,1.0	180,1.8	90,1.0	90,-

Notes:(1) In [6], rate 2/3, 3/4, 4/5, 5/6 are generated by discarding several systematic bits and/or parity bits from rate 1/2 outputs, which is also called the puncturing process. In addition, in [6] only 50% of the messages are stored in the memory, so the total number of bits should be 105 kb

of other decoders. The energy efficiency and the hardware efficiency are also at the same level as the best state-of-the-art techniques. Although the latency of the proposed decoder is higher, this is because we divide all the updating operations for one layer into three clock cycles due to the constraints of the reading/writing operations of posterior messages.

VI. CONCLUSION

In this brief, we have presented an efficient multirate LDPC-CC decoder for the IEEE 1901 standard using the layered decoding algorithm. From the simulation results, it can be seen that the layered decoding algorithm can achieve a better performance than the original MP algorithm with only half the number of processors. A new architecture has been developed, and a decoder that can support all the code rates (1/2, 2/3, 3/4, 4/5) for LDPC-CCs defined in the IEEE 1901 standard has been fabricated and measured. To the best of our knowledge, IEEE 1901 is the first communication standard that adopts LDPC-CC code, and this decoder is the first LDPC-CC decoder that supports a communication standard. The results show that the proposed decoder is both memory and area efficient.

REFERENCES

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

[2] 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, Sep. 1999.

[3] D. Chang *et al.*, "LDPC convolutional codes using layered decoding algorithm for high speed coherent optical transmission," presented at the Optical Fiber Communication Conference, Los Angeles, CA, USA, 2012, Paper OW1H.4.

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

[5] J. Sha, Z. Wang, M. Gao, and L. Li, "Multi-Gb/s LDPC code design and implementation.," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 2, pp. 262–268, Feb. 2009.

[6] C.-L. Chen, Y.-H. Lin, H.-C. Chang, and C.-Y. Lee, "A 2.37-Gb/s 284.8 mW rate-compatible (491, 3, 6) LDPC-CC decoder," *IEEE J. Solid-State Circuits*, vol. 47, no. 4, pp. 817–831, Apr. 2012.

[7] Z. Chen *et al.*, "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.

[8] R. Swamy *et al.*, "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.

[9] T. L. Brandon *et al.*, "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. 56, no. 5, pp. 1017–1029, May 2009.