A Reduced-Complexity Architecture for LDPC Layered Decoding Schemes

Sangmin Kim, Gerald E. Sobelman, and Hanho Lee

***Abstract*—A reduced-complexity low density parity check (LDPC) layered decoding architecture is proposed using an offset permutation scheme in the switch networks. This method requires only one shuffle network, rather than the two shuffle networks which are used in conventional designs. In addition, we use a block parallel decoding scheme by suitably mapping between required memory banks and processing units in order to increase the decoding throughput. The proposed architecture is realized for a 672-bit, rate-1/2 irregular LDPC code on a Xilinx Virtex-4 FPGA device. The design achieves an information throughput of 822 Mb/s at a clock speed of 335 MHz with a maximum of 8 iterations.**

***Index Terms*—Decoding, field-programmable gate array (FPGA), forward error correction, low density parity check (LDPC).**

I. Introduction

Low density parity check (LDPC) codes have been extensively adopted in next-generation forward error correction applications because they achieve very good performance using the iterative decoding approach of the belief-propagation (BP) [1]. The basic decoder design [2] for achieving the highest decoding throughput is to allocate processors corresponding to all check and variable nodes, together with an interconnection network. In this fully-parallel decoder architecture, the hardware complexity due to the routing overhead is very large. Therefore, much of the work on LDPC decoder design has been directed towards achieving optimal tradeoffs between hardware complexity and decoding throughput. In particular, a time-multiplexed or folded approach [3], which is known as a partially parallel decoder architecture, has been proposed.

Recently, several partially parallel decoder designs [4]–[10] for "block structured LDPC codes" or "architecture-aware LDPC codes" have been developed using elements such as check node units (CNUs), variable node units (VNUs), and interconnection networks between CNUs and VNUs. These approaches lead to decoders having a reduced number of clock cycles per iteration, which results in higher decoding throughput. In [5], the sum and sign accumulation unit for the CNU is used in computing a portion of each row while the VNU computes each column. The overlapped decoding scheme exploited in [6] for high-rate LDPC codes is similar to the method in [5] except that a CNU computes a portion of a row by accumulating partial results. To achieve a faster convergence compared to the overlapped, two-phase decoding scheme, turbo-decoding message-passing [7], or a layered decoding [8] schedule and architecture for regular structured codes have been proposed. However, conventional layered decoders use a bidirectional network or two switch networks for shuffling and reshuffling messages, which increases the hardware complexity.

Designs utilizing one data shifter or a cyclic shifter have been introduced in [9] and [10], respectively. However, the proposed data shifter in [9] is targeted for only one parity-check matrix $\mathbf{H}$ since its interconnection mapping is fixed by the shift values in the first block row of $\mathbf{H}$. In [10], the overall decoder is designed specifically for a (3, 6) array code. In contrast, our objective is to create a design that is suitable for multiple code rates and different codeword sizes. Specifically, we propose a reduced-complexity LDPC decoder architecture for use in layered decoding having an offset generating algorithm to decrease the interconnection complexity with no degradation in the decoding throughput.

II. Layered Block Parallel Decoder Architecture

A. Layered Decoding Scheme

Structured regular or irregular LDPC codes are described by an $M_\mathrm{b} \times N_\mathrm{b}$ base matrix $\mathbf{H}_\mathrm{b}$ with $M_\mathrm{b} = M/z$ and $N_\mathrm{b} = N/z$, where M is the number of parity check equations, N is the code length, and z is the size of a square sub-matrix. The parity check matrix $\mathbf{H}$ of a structured LDPC code can be viewed as the concatenation of constituent codes [7], where the number of constituent codes is equal to M_b. The dataflow of a typical layered decoder is shown in Fig. 1. Let $\mathrm{R} = [\mathbf{r}_1 \mathbf{r}_2 \cdots \mathbf{r}_{M_\mathrm{b}}]^T$ denote the check-to-variable messages, where $\mathbf{r}_k$ corresponds to a constituent code of $\mathbf{H}$ for $1 \le k \le M_\mathrm{b}$. $\mathbf{Q}^{(k)}$ and $\mathbf{Q}^{(k+1)}$ are the previously decoded soft output value and the newly decoded soft output value used for updating the next block row, respectively. $\mathbf{L}^{(k)}$ denotes the variable-to-check message which has entered the decoding update block, and $\mathbf{r}_k^+$ represents the updated check-to-variable message at the kth block row. The updated check-to-variable message $\mathbf{r}_k^+$ can be expressed as [8]

$$\mathbf{r}_k^+ = -\prod \mathrm{sign}(\mathbf{L}^{(k)}) \cdot \Psi \left\{ \sum \Psi \left(\mathbf{L}^{(k)} \right) \right\}$$

where

$$\Psi(x) = \log \left(\tanh \left(\left| \frac{x}{2} \right| \right) \right). \tag{1}$$

For notational simplicity, we omit the indices denoting the set of positions of the columns connected to all check nodes within the kth block row. In Fig. 1, the decoding update block, which was presented as a check node-based processor (CNBP) in [11], can be implemented for any decoding algorithm such as approximations of BP.

After the initialization of the layered decoder is achieved using the soft values from the channel in the bit update block, the decoder starts updating messages corresponding to the first constituent code ($\mathbf{r}_1$). The switch network (SN) 1 shuffles the channel soft values based on the permutation information obtained from $\mathbf{r}_1$. The shifted messages $\mathbf{Q}^{(1)}$ from SN 1 and the check-to-variable messages $\mathbf{r}_1$ read from memory are used to compute the variable-to-check messages $\mathbf{L}^{(1)}$. The decoding update block computes the check-to-variable messages $\mathbf{r}_1^+$ based on $\mathbf{L}^{(1)}$ and stores $\mathbf{r}_1^+$ back into memory. The updated posterior messages are computed by adding the recently updated check-to-variable messages to the variable-to-check messages, then reshuffled through SN 2 and finally stored as $\mathbf{Q}^{(2)}$ in the bit update

Manuscript received August 15, 2009; revised November 18, 2009. First published March 29, 2010; current version published May 25, 2011. This work was supported in part by the Electronics and Telecommunications Research Institute (ETRI).

S. Kim and G. E. Sobelman are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: sangmin@ece.umn.edu; sobelman@umn.edu).

H. Lee is with the School of Information and Communication Engineering, Inha University, Incheon 402-751, Korea (e-mail: hhlee@inha.ac.kr).

Digital Object Identifier 10.1109/TVLSI.2010.2043965

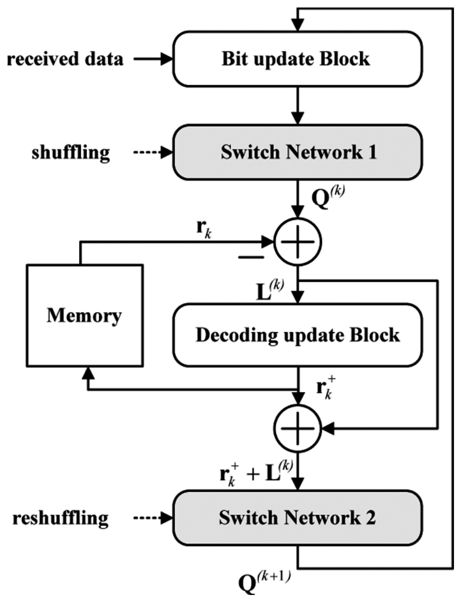

Fig. 1. Dataflow of a typical layered decoder.

block. This updated soft output value $\mathbf{Q}^{(2)}$ is used to compute messages corresponding to the next constituent code $(\mathbf{r}_2)$. Decoding for a constituent code $(\mathbf{r}_k)$ or for the complete $\mathbf{H}$ is called one sub-iteration or one iteration, respectively.

B. Block Parallel Decoder Architecture

Compared to the decoder structures presented in [5]–[8], this decoder architecture has the following unique characteristics: 1) we generate offset shifting values for shuffling and reshuffling messages so that the proposed decoder needs to use only SN 1 rather than two SNs and 2) the number of memory banks for check-to-variable messages is configured to be a row weight of $\mathbf{H}$. The first characteristic is achieved by observing that the operation of the SNs for shuffling and reshuffling messages is overlapped during the updating of the constituent codes of $\mathbf{H}$. In other words, the SN 2 block in Fig. 1 reshuffles updated output messages corresponding to $\mathbf{r}_k$ until the decoder reaches the end of one iteration for the complete $\mathbf{H}$. At the end of a sub-iteration the recently updated outputs are shuffled by SN 1 for the next constituent code. Therefore, the two consecutive operations, reshuffling and shuffling, are not necessary to compute the decoded output within a sub-iteration and this provides an opportunity for reducing the complexity of the interconnections. The second characteristic is used to simultaneously process all messages corresponding to $\mathbf{r}_k$ in one clock cycle.

The dataflow of the layered decoder architecture based on the above two characteristics is shown in Fig. 2(a). The decoding steps are almost the same as in the conventional decoding with the exception of the ordering patterns in the bit update block and the offset permutations through SN 1. Let $\mathbf{P}^{(k)}$ and $\mathbf{P}^{(k+1)}$ be the previous and updated soft outputs, respectively. Note that $\mathbf{P}^{(k+1)} = \prod(\mathbf{Q}^{(k+1)})$ is a permutation of $\mathbf{Q}^{(k+1)}$. The top-level architecture using the layered mode with offset permutations for SN 1 is illustrated in Fig. 2(b). During an initialization operation, the incoming soft message is shifted into the bit updating register array. Then, the registered MUX block simultaneously loads the required messages into SN 1. Following that, SN 1 rotates the input messages by the amount of the offset permutations. The check-to-variable messages and the rotated variable messages loaded into the CNBP blocks are then computed for newly updated check-to-variable messages and rotated soft output messages. The check-to-variable messages are stored in the memory, and the rotated soft output messages replace the previous messages in the bit-update register array.

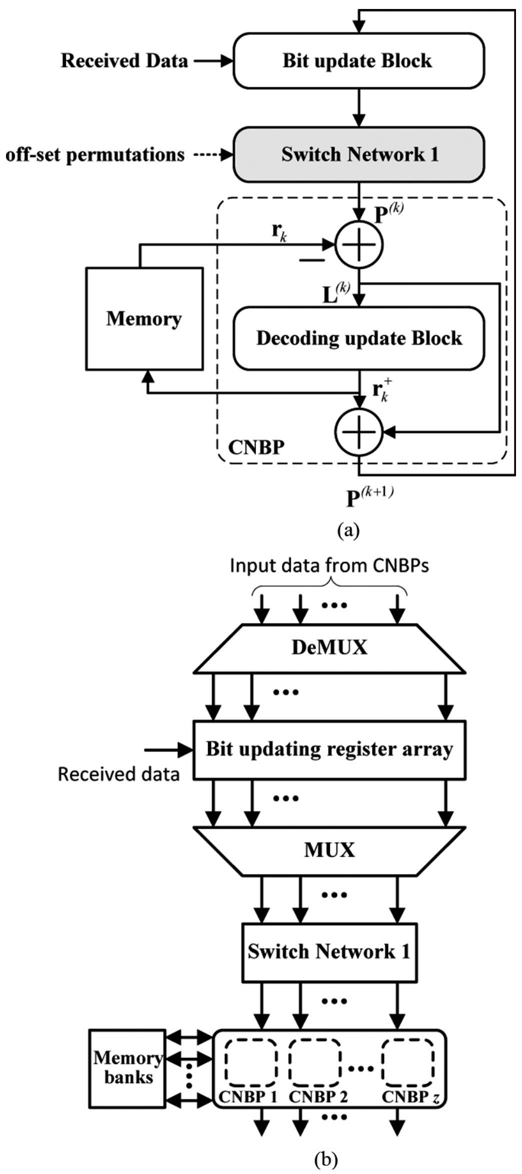

Fig. 2. Proposed layered decoder. (a) Modified dataflow with offset permutations. (b) Block parallel layered decoder architecture.

III. Algorithm for Generating Offset Values of Switch Network

We present a novel algorithm to generate offset values for switch networks which leads to the elimination of SN 2. A decoder design using this proposed algorithm decreases the hardware cost by removing redundant shifting operations.

In general, indices of the base matrix $\mathbf{H}_{\rm b} = (h_{m,n})$ are usually represented by cyclically shifting the columns of the identity matrix $\mathbf{I}_{z\times z}$ to the right or left by $h_{m,n}$ places, where $h_{m,n} \in \{0, 1, \ldots, z-1\} \cup \{-1\}$, for $m = 1, \ldots, M_{\rm b}$ and $n = 1, \ldots, N_{\rm b}$, in which "-1" represents null (i.e., all-zero) submatrices. We denote two $M_{\rm b} \times N_{\rm b}$

matrices of precomputed cyclic shifts for SN 1 and 2 as $\mathbf{A} = (a_{m,n})$ and $\mathbf{B} = (b_{m,n})$, respectively, where $a_{m,n}, b_{m,n} \in \{0, 1, \ldots, z-1\} \cup \{-1\}$, for $m = 1, \ldots, M_{\mathrm{b}}$ and $n = 1, \ldots, N_{\mathrm{b}}$. The required cyclic shifting values of $\mathbf{A}$ and $\mathbf{B}$ can be set according to either shuffling or reshuffling operations for SN 1 or SN 2, respectively. All integer elements i, where $i \in \{0, 1, \ldots, z-1\}$, of $\mathbf{A}$ and $\mathbf{B}$ can be stored in a dedicated lookup table (LUT). The proposed algorithm for generating offset shifting values can be described as follows.

Algorithm 1

```
for $n = 1 : N_{\mathrm{b}}$
    $m \leftarrow 1$
    while $a_{m,n} == -1$
        $s_{m,n} \leftarrow -1$
        $m \leftarrow m + 1$
    end
    $s_{m,n} \leftarrow a_{m,n}$
    $m \leftarrow m + 1$
    while $m \leq M_{\mathrm{b}}$
        $l \leftarrow m - 1$
        while $b_{l,n} == -1$ and $l \geq 1$
            $l \leftarrow l - 1$
        end
        $s_{m,n} = a_{m,n} \bigoplus b_{l,n}$
        $m \leftarrow m + 1$
    end
end
```

where

$$a \oplus b = \begin{cases} -1, & a = -1 \text{ or } b = -1 \\ (a+b) \bmod \mathbf{z}, & \text{otherwise.} \end{cases}$$

In the above, we indicate that each element $s_{m,n}$ of the matrix $\mathbf{S} = (s_{m,n})$ is an offset shifting value. Therefore, we need only use SN 1 with offset shifting information $s_{m,n}$, which exploits the characteristic structure of the layered decoding scheme. Based on a given set of $s_{m,n}$ shifting values, we can reduce the amount of hardware required by removing any unnecessary 2×2 switches from the switching network. To compute the parity check equations using the hard decoded output $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$, which is the same as determining if $\mathbf{H} \cdot \mathbf{x} = \mathbf{0}$, the shifting information for performing the parity check equations and for sorting correctly the hard decision output is needed after each iteration. The shifting information $\mathbf{D} = (d_n)$, for $n = 1, \ldots, N_{\mathrm{b}}$, is described in Algorithm 2.

Algorithm 2

```
for $n = 1 : N_{\mathrm{b}}$
    $m \leftarrow M_{\mathrm{b}}$
    $d_n \leftarrow b_{m,n}$
    while $d_n == -1$
        $m \leftarrow m - 1$
        $d_n \leftarrow b_{m,n}$
    end
end
```

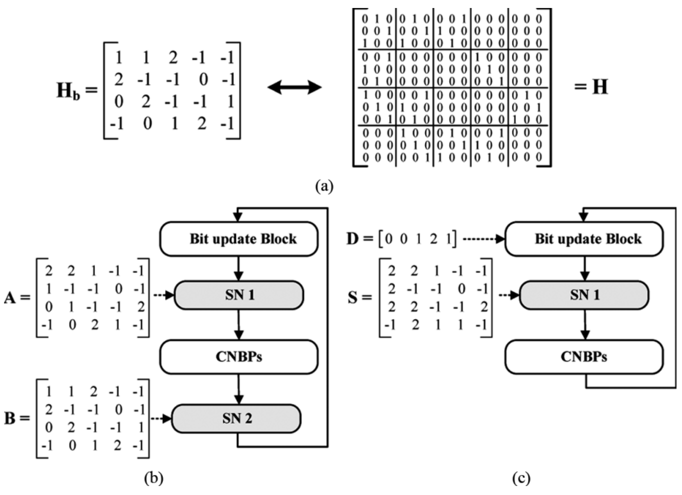

Fig. 3. Example of generating offset values for the switch networks. (a) 4×5 $\mathbf{H}_{\mathrm{b}}$. (b) Typical layered decoder. (c) Proposed decoder.

Given the shifting information $\mathbf{D}$ and the hard decisions x, we can precompute the output ordering information $\mathbf{y} = \mathbf{E} \cdot \mathbf{x}$, where $\mathbf{E}$ can be written as

$$\mathbf{E} = \begin{bmatrix} \boldsymbol{p}^{d_1} & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \boldsymbol{p}^{d_2} & \ddots & \vdots \\ \vdots & & & \mathbf{O} \\ \mathbf{O} & \cdots & \mathbf{O} & \boldsymbol{p}^{d_{N_{\mathrm{b}}}} \end{bmatrix}.$$

Here, $\mathbf{O}$ indicates a $z \times z$ zero matrix, and $\boldsymbol{p}^j$ is obtained from the $\mathbf{I}_{z \times z}$ by cyclically shifting the columns to the right by j elements. From the output ordering information $\mathbf{y}$, the decoded data can be easily mapped to the output ports of this decoder without extra hardware cost.

1) Example: The base matrix $\mathbf{H}_{\mathrm{b}}$ in Fig. 3(a) is a 4×5 array of 3×3 cyclicly-shifted identity or all-zero matrices. The control signals represented in matrix form, $\mathbf{A}$ and $\mathbf{B}$, for SN 1 and SN 2 as shown in Fig. 3(b) can be determined based on the elements of the $\mathbf{H}_{\mathrm{b}}$. We compute offset control signals and decoded output mapping information, which are illustrated in Fig. 3(c), by using the proposed Algorithms 1 and 2.

IV. Hardware Complexity Comparison

In layered decoding architectures the number of memory bits is reduced by nearly 50% and the number of iterations for achieving the same error rate is also reduced by almost 50% compared with traditional decoder designs [8]. To show the low complexity of the block parallel processor in the layered decoding scheme, we compare it with different decoder architectures.

The irregular LDPC code in the IEEE 802.15 standard has a 16 by 32 $\mathbf{H}_{\mathrm{b}}$ with $z = 21$, so that its parity check matrix $\mathbf{H}$ has 16×21 rows and 32×21 columns. Tables I and II present, for three different designs, characteristics of the key components used and the estimated total number of hardware resources required, respectively (note that designs A [4] and B [5] do not use layered decoding). In Table I, design A is obtained using a folding factor of 4 for both the CNUs and the VNUs in a time-multiplexed approach and it requires eight clock cycles to complete one decoding iteration. Design B uses a set of sum and sign accumulation units (SSAUs) in addition to the CNUs and VNUs. In this design, the CNUs and SSAUs are fully parallel while the VNUs have

TABLE I
KEY COMPONENT CHARACTERISTICS FOR THREE DIFFERENT DESIGNS

	Design A [4]		Design B [5]			Proposed scheme
	CNU	VNU	CNU	VNU	SSAU	CNBP
LUT	8	4	1	4	1	16
Adder	15	8	1	8	1	31
Ex-OR	15	-	1	-	1	15
SM-2's	-	4	-	4	-	8
2's-SM	-	4	1	-	1	8
Registers	-	-	-	-	2	25
No. of functional unit	84	168	336	84	336	21

TABLE II
ESTIMATED TOTAL HARDWARE RESOURCES FOR THREE DIFFERENT DESIGNS

	Design A [4]	Design B [5]	Proposed scheme
LUT	1,344	1,008 (100%)	336 (33%)
Adder	2,604	1,344 (100%)	651 (48%)
Ex-OR	1,260	672 (100%)	315 (47%)
SM-2's	672	336 (100%)	168 (50%)
2's-SM	672	336 (100%)	168 (50%)
Registers	-	672 (100%)	525 (78%)

a folding factor of 8, and it requires nine clock cycles to complete one decoding iteration.

To perform a fair comparison with the hardware complexity of designs A and B, we use a standard Log-BP structure in the CNBP as shown in Fig. 4. For simplicity, sign-magnitude (SM), 2's complement (2's) and exclusive-or (XOR) units are not shown in the figure. Pipeline registers are inserted to provide the same critical path in all 3 designs, which is equal to the path from a LUT to an eight-input adder tree block. For the $\mathbf{H}$ matrix considered here, there are no data dependencies between adjacent layers while updating posterior messages. For other $\mathbf{H}$ matrices having such dependencies, stalls could be used to avoid conflicts in the pipeline.

The decoding throughput can be approximated as

$$\text{Throughput} \approx \frac{N \times f_{\text{clk}} \times R}{N_{\text{clk}} \times N_{\text{iter}} + N_{\text{latency}}} \quad (2)$$

where f_{clk} is the clock frequency, R is the code rate, N_{clk} is the number of clock cycles required for an iteration, N_{iter} is the average number of iterations, and N_{latency} is the number of clock cycles due to the pipeline latency. Note that a layered decoding scheme needs only about half the average number of iterations compared with designs A and B in order to achieve same error rate. Therefore, the proposed design uses about twice as many clock cycles per iteration (i.e., 16 clock cycles versus 8 for design A and 9 for design B) with no throughput degradation. As shown in Table II, the hardware complexity of the decoding processing units and the amount of memory required for the proposed design is significantly smaller than for either design A or B.

V. IMPLEMENTATION RESULTS

We designed an $N = 672$ (data length $= 336$), rate-1/2 irregular LDPC decoder based on the proposed offset control scheme for the SN using a block parallel architecture. The min-sum decoding algorithm, which is a modified version of the standard Log-BP algorithm, is exploited in the CNBP, which has four pipeline stages. Based on the simulated performance results of Fig. 5(a), we use a $(q, f) = (6, 2)$ quantization scheme, where q and f are the total bit size and the number of fractional bits, respectively. Furthermore, our decoder typically needs only three iterations to converge at a signal-to-noise ratio of 3 dB, as shown in Fig. 5(b).

Our SN uses a Benes network [13] in which the unnecessary switches have been removed, and it uses three pipeline stages in order to reduce the critical path delay. As a result, the critical path of the pipelined SN is three 2×2 switches. This decoder was implemented on the Xilinx

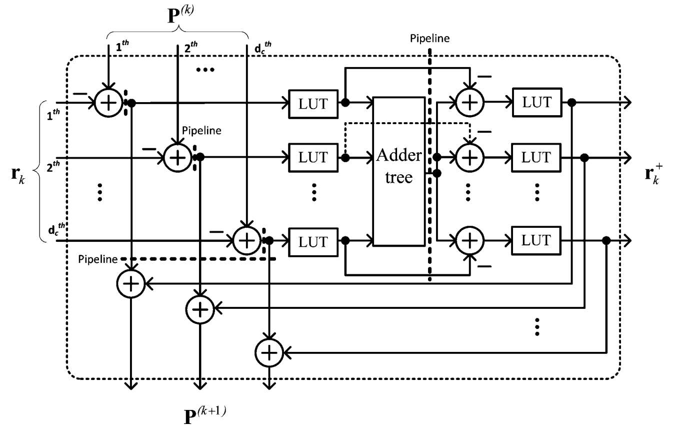

Fig. 4. Simplified diagram of a block parallel layered decoding unit, the CNBP.

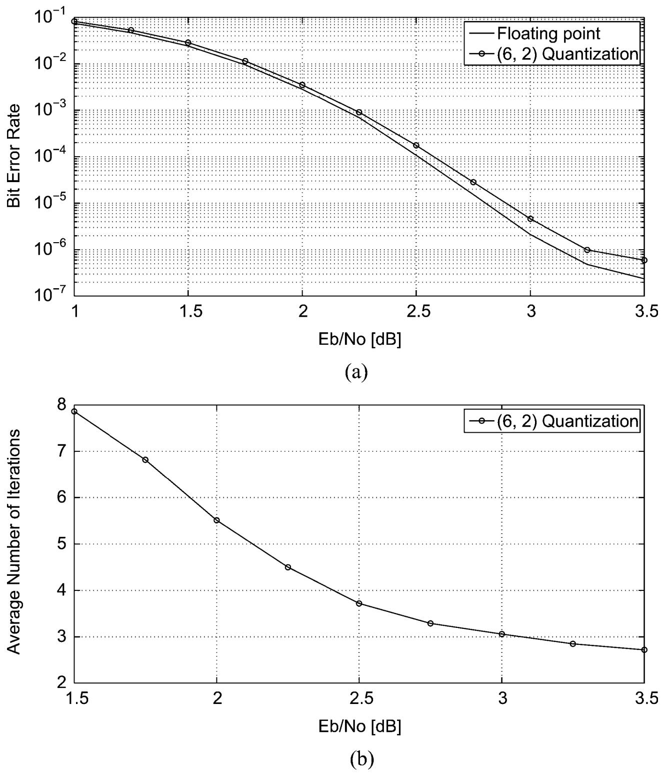

Fig. 5. Simulated performance for $N = 672$, rate-1/2 irregular LDPC code. (Maximum number of iterations $= 8$). (a) Bit error rate for the proposed design. (b) Average number of iterations using the (6, 2) quantization.

TABLE III
XILINX VIRTEX4 XC4VLS200 FPGA SYNTHESIS RESULTS

Resource	Conventional layered decoding	Proposed scheme	Improvement
Slices	29.763	27,003	9.27%
Slice Flip Flops	26,281	23,206	11.7%
4 input LUTS	51,298	45,409	11.5%
Block RAMs	32	32	-
Throughput	798 Mb/s	822 Mb/s	3.01%

Virtex-4 xc4vls200 FPGA. To provide a fair comparison, we also implemented a conventional layered decoding design for the same code using the same quantization and pipelining techniques. The synthesis results for both designs are given in Table III. The proposed decoder, using only a single SN, results in 9.3% reduction in the number of slices with no degradation in the decoding throughput.

The proposed decoder has a pipeline latency of nine clock cycles, of which seven cycles are due to the SN 1 and CNBP blocks and the other two cycles are due to the registered MUX and DEMUX blocks. The information decoding throughput is estimated to be approximately $(335\text{ MHz}) \times 336/(8 \times 16 + 9) = 822$ Mb/s based on the maximum clock frequency of 335 MHz (from the synthesis timing report) and using a maximum of eight decoding iterations and the pipeline latency of nine cycles. The estimated gate count (14 adders, 20 muxes, and 11 XOR gates per VNU and CNU) in [9] is based on a different code, i.e., a 3456-bit, rate 1/2, (3, 6)-regular code, i.e., having a column weight of 3 and a row weight of 6. The design in [9] requires 12 clock cycles per iteration, while our decoder design requires only three clock cycles per iteration. However, the estimated gate count of our design (21 adders, 31 muxes, and 11 XOR gates) is higher than that of [9].

VI. CONCLUSION

We have proposed an efficient architecture for layered LDPC decoding by reducing the interconnection complexity without any degradation in the decoding throughput. Our design requires only a single shuffle network, rather than the two shuffle networks used in prior designs. The results show a 9.3% reduction in the number of required FPGA slices compared to a standard layered decoding architecture. Implementation of a 672-bit, rate-1/2 irregular LDPC code on a Xilinx Virtex-4 xc4vls200 FPGA device achieves an information throughput of 822 Mb/s with a maximum of eight iterations.

REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[2] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002.

[3] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.

[4] T. Zhang and K. K. Parhi, "Joint (3,k)-regular LDPC code and decoder/encoder design," *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1065–1079, Apr. 2004.

[5] S. Kim, K. K. Parhi, and R. Liu, "System and method for designing RS-based LDPC code decoder," U.S. Patent App. US2007-0033484, Feb. 8, 2007.

[6] L. Liu and C.-J. R. Shi, "Sliced message passing: High throughput overlapped decoding of high-rate low-density parity-check codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3697–3710, Nov. 2008.

[7] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 684–698, Mar. 2006.

[8] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop Signal Process. Syst. (SIPS)*, Austin, TX, Oct. 2004, pp. 107–112.

[9] Z. Cui, Z. Wang, and Y. Liu, "High-throughput layered LDPC decoding architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 4, pp. 582–587, Apr. 2009.

[10] K. K. Gunnam, G. S. Cho, and M. B. Yeary, "A parallel VLSI architecture for layered decoding for array LDPC codes," in *Proc. Int. Conf. VLSI Des.*, Bangalore, India, Jan. 2007, pp. 738–743.

[11] S. Kim, G. E. Sobelman, and H. Lee, "Flexible LDPC decoder architecture for high-throughput applications," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Macao, China, Nov. 2008, pp. 45–48.

[12] *Draft Standard for Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs): Amendment 2: Millimeter-Wave Based Alternative Physical Layer Extension*, IEEE P802.15.3c/D04, 2008. [Online]. Available: http://standards.ieee.org

[13] S. Olcer, "Decoder architecture for arrary-code-based LDPC codes," in *Proc. IEEE Global Telecommun. (GLOBECOM) Conf.*, San Francisco, CA, Dec. 2003, pp. 2046–2050.