# VLSI Architectures for Layered Decoding for Irregular LDPC Codes of IEEE 802.11n

Kiran Gunnam, Weihuang Wang, Gwan Choi
Department of Electrical and Computer Engineering
Texas A&M University, TX 77840
Email: {kirang, whwang, gchoi}@ece.tamu.edu

Mark Yeary
Department of Electrical and Computer Engineering
University of Oklahoma, Norman, OK 73109
Email: yeary@ou.edu

***Abstract*— We present a new multi-rate architecture for decoding block LDPC codes in IEEE 802.11n standard. The proposed architecture utilizes the value-reuse property of offset min-sum, block-serial scheduling of computations and turbo decoding message passing algorithm. Techniques of data-forwarding and out-of-order processing are used to deal with the irregularity of the codes. The decoder has the following advantages when compared to recent state-of-the-art architectures: 55% savings in memory, reduction of routers by 50% and increase of throughput by 2x.**

## I. Introduction

Low-Density Parity-Check (LDPC) codes and Turbo codes are among the known near Shannon limit codes that can achieve very low bit error rates for low signal-to-noise ratio (SNR) applications [1]. When compared to that of Turbo codes, LDPC decoding algorithm has more parallelization, low implementation complexity, low decoding latency, as well as no error-floors at high SNRs. LDPC codes are considered for virtually all the next generation communication standards.

LDPC codes can be decoded by Gallager's iterative two phase message passing algorithm (TPMP) which involves check-node update and variable-node update as two phase schedule. Widely used algorithms are sum of products (SP), min-sum (MS) and Jacobian based BCJR (named after its discoverers Bahl, Cocke, Jelinik and Raviv). Chen *et al.* [2] showed that the offset min-sum (OMS) decoding algorithm with 5-bit uniform quantization could achieve the same bit-error rate (BER) performance as that of floating point SP and BCJR with less than 0.1 dB SNR penalty. In addition, the authors in [3] introduced the concept of turbo decoding message passing (TDMP, also called as layered decoding) for their AA-LDPC codes using BCJR, which is able to reduce the number of iterations required by up to 50% without performance degradation when compared to the standard message passing algorithm. TDMP is later studied and applied for different LDPC codes using SP and its variations[4], [5].

While fully-parallel LDPC decoder designs [6] suffered from complex interconnect issues, various semi-parallel implementations based on structured LDPC codes [3], [7], [8], [9], [10], [11] alleviate the interconnect complexity. Block codes [13] are among the family of structured LDPC codes, proposed for IEEE 802.11n, IEEE 802.16e [12], and IEEE 802.22. As shown in Fig. 1, the entire $H$ matrix of block LDPC is composed of either an identity matrix with different cyclic shifts (represented as a "1" in the figure) or null matrix (represented as a "0"). The expansion factor, defined as size of the identity matrix $z$ can be 27, 54 or 81 [13]. All the base matrices have the same number of block columns $N_b = 24$, and the code length $N$ is $N_b \times z$. The decoder supports all the code lengths (648, 1296 and 1944) and all the code rates ($1/2, 2/3, 3/4$ and $5/6$) as specified in IEEE 802.11n standard [13].

The main contribution of this work is an efficient architecture which utilizes the value-reuse property of OMS, cyclic shift property

$$\begin{bmatrix}
1&1&1&1&1&1&1&0&1&1&0&0&1&0&0&1&1&0&0&0&0&0&0&1\\
1&1&1&1&1&1&0&1&0&0&0&0&0&0&1&0&1&1&0&0&0&0&0&0\\
0&1&1&1&1&1&1&0&0&0&1&1&0&0&0&1&0&1&1&0&0&0&0&0\\
0&0&1&1&1&1&0&0&1&1&0&0&1&1&0&0&0&0&1&1&0&0&0&0\\
0&0&0&1&1&1&0&1&0&0&1&1&0&0&1&0&0&0&0&1&1&0&0&0\\
0&0&0&0&1&1&1&0&1&0&0&0&1&1&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&1&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&1&1&0&0&0&0&0&0&0&1&1
\end{bmatrix}$$

Fig. 1. Example of base matrix of block LDPC, rate $2/3$ base matrix for IEEE 802.11n.

of structured LDPC codes and block serial scheduling for block LDPC codes in IEEE802.11n. With minor changes, this architecture can be applied to codes in IEEE 802.16e and IEEE 802.22. The resulting decoder architecture has significantly lower requirements of logic, interconnection and memory as compared with recent state-of-the-art implementations [3], [5], [8], [9], [11], [15]. The rest of the paper is organized as follows. Section II introduces min-sum decoding algorithm and TDMP. Section III presents the value-reuse property and new micro-architecture structure for check-node units (CNU). The data flow graph and architecture for TDMP using OMS are described in Section IV while Section V presents the FPGA and ASIC implementation results and discussion. Finally, Section VI concludes the paper.

## II. LDPC decoding

The reliability messages used in TPMP based the OMS algorithm can be computed in two phases, viz. check-node processing and variable-node processing. The two operations are repeated iteratively until the stop criterion is satisfied. This is also referred to as standard message passing or two phase message passing (TPMP). Denote $M(n)$ as the set of neighboring check nodes for variable node $n$ and $N(m)$ as the set of neighboring variable nodes for check node $m$. For the $i^{th}$ iteration, $Q^i_{nm}$ is the message from variable node $n$ to check node $m$, $R^i_{mn}$ is the message from check node $m$ to variable node $n$. The check-node process can be represented as in (1)-(3): $\forall m \in M(n)$ and $n \in N(m)$,

$$R^i_{mn} = \delta^i_{mn} max\left(\kappa^i_{mn} - \beta, 0\right) \tag{1}$$

$$\kappa^i_{mn} = \left|R^i_{mn}\right| = \min_{n' \in N(m)} \left|Q^{i-1}_{n'm}\right| \tag{2}$$

$$\delta^i_{mn} = \prod_{n' \in N(m)\backslash n} sign\left(Q^{i-1}_{n'm}\right) \tag{3}$$

where $\beta$ is a positive constant and depends on the code parameter, $\delta^i_{mn}$ is the sign of $R^i_{mn}$ and $n' \in N(m)\backslash n$ means all the check nodes connected with variable node $m$ except check node $n$. We used the density evolution procedure in [2] to compute $\beta$ for the

different codes supported in IEEE 802.11n standard. In variable-node processing, for each $n$ and $m \in M(n)$, messages are defined as

$$Q_{nm}^{i} = L_{n}^{0} + \sum_{m' \in M(n) \backslash m} R_{m'n}^{i} \tag{4}$$

where $L_n^0$ equal to the received channel value is the log-likelihood ratio of bit $n$. After variable-node processing in each iteration, the $P$ values as computed in Eq. 5

$$P_n = L_n^i + \sum_{m \in M(n)} R_{mn}^i. \tag{5}$$

A hard decision is taken where $\hat{x}_n = 0$ if $P_n(x_n) \geq 0$, $\hat{x}_n = 1$ if $P_n(x_n) < 0$. If $\hat{x}H^T = 0$, the decoding process is finished with $\hat{x}$ as the decoder output; otherwise, go to check-node processing. If the decoding process doesn't end within some maximum number of iterations $it_{max}$, stop and output an error message.

In TDMP, the block LDPC with $j$ block rows can be viewed as a concatenation of $j$ layers or constituent sub-codes similar to observations made for AA-LDPC codes in [3]. After the check-node processing is finished for one block row, the messages are immediately used to update the variable nodes, whose results are then provided for processing the next block row of check nodes. This differs from TPMP where all check nodes are processed first and then the variable node messages will be computed. Each decoding iteration in the TDMP is composed of $j$ number of sub-iterations. In the beginning of the decoding process, $P$ messages and check-node messages are initialized to channel values and zero respectively:

$$\vec{P}_m = \vec{\Lambda}_m, \vec{R}_{l,m}^0 = 0, \quad \forall m \in 1, 2, \cdots N_b, \forall l \in 1, 2, \cdots j \tag{6}$$

where subscript $m$ represent the $m^{th}$ block column. Variable messages are processed as $\forall l = 1, 2, \cdots j, \forall n = 1, 2, \cdots k$

$$\left[\vec{Q}_{l,n}^{i}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)} - \vec{R}_{l,n}^{i-1} \tag{7}$$

where subscript $l$ means $l^{th}$ block row, $k_l$ is check node degree of $l^{th}$ block row, $S(l,n)$ denotes the shift coefficient for the $l^{th}$ block row and $n^{th}$ non-zero block of the matrix (note that null blocks in H matrix need not be processed). After completion of each block row, variable messages are immediately used to update the new check node messages:

$$\vec{R}_{l,n} = f(\left[\vec{Q}_{l,n'}^{i}\right]^{S(l,n')}), \quad \forall n' = 1, 2, \cdots, k_l. \tag{8}$$

Here $[]^{S(l,n)}$ denotes that the vector is cyclically shifted up by the amount $S(l,n)$. $f(\cdot)$ denotes the check-node processing. For the proposed work we use OMS as defined in (1)-(3).

$$\left[\vec{P}_n^{i}\right]^{S(l,n)} = \left[\vec{Q}_{l,n}^{i}\right]^{S(l,n)} + \vec{R}_{l,n}^{i} \tag{9}$$

The completion of check-node processing and associated variable-node processing (7-9)of all block rows constitutes one iteration. Convergence is achieved when $\hat{x}H^T = 0$ for all the sub-iterations in an iteration and $\hat{x}$ does not change throughout the iteration.

## III. Value-reuse Properties of OMS

This section presents the micro-architecture of serial CNU for OMS, which was used in our recent work on TPMP architecture [10], [14]. For each check node $m$, $\left|R_{mn}^{i}\right|, \forall n \in N(m)$ takes only 2 values. In addition, since $\delta_{mn}^{i}, \forall n \in N(m)$ is either $+1$ or $-1$. Equation (1) gives rise to only 3 possible values for the whole set $R_{mn}^{i}, \forall n \in N(m)$.

Fig. 2(a) shows micro-architecture of a serial CNU. M1_M2

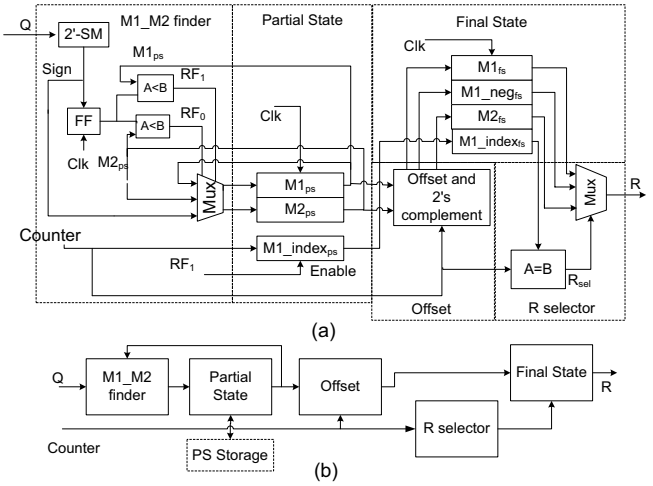


Fig. 2. (a) Micro-architecture of CNU. (b) Block diagram of CNU.

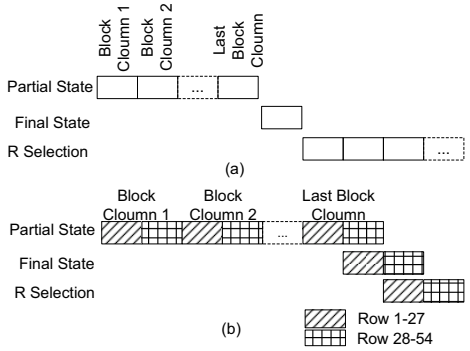


Fig. 3. Operation of CNU (a) no time-division multiplexing (b) time-division multiplexing.

finder computes the two least numbers, $M1$ and $M2$ respectively. $M1$, $M2$ and index of $M1$ are stored in partial state. The offset module applies the offset correction, and stores the results in the Final State module. It should be noted that the final state includes only $M1$,$-M1$, $\pm M2$ with offset correction and index of $M1$. R Selector then selects the output $R$ messages based on the index comparison and sign of $R$ as in 3. In operation, the Partial State, Final State and R selection will operate on different check nodes simultaneously. Normally CNU processing uses the signed magnitude arithmetic for (1)-(2) and VNU processing (4)-(5) uses 2's complement arithmetic. 2's complement is converted to signed magnitude at the inputs of CNU and converted back to 2's complement at the output of CNU. Fig. 2(b) shows the block diagram for the CNU, the PS storage element is necessary when one CNU is time-division multiplexed to operate on multiple check-nodes. The operation of the CNU is explained in Fig. 3, where each CNU is dedicated to one check node processing, in the first $k$ number of clock cycles of the check-node processing, incoming variable messages are compared with the two up-to-date least minimum numbers(PS) to generate the new partial state. The final state (FS) is executed after finishing all incoming variable-node messages. Output of the $R$ message is able to begin at the completion of final state computation. When one CNU is used to operate on multiple check-nodes (2, for example), the execution is outlined in Fig. 3(b). Incoming variable-messages are interleaved, so is operation of M1_M2 finder, with the help of storage element for partial state. It should be noted while the partial state is always operating in order, the R selection can be out of order for block LDPC decoding, as explained later in Section IV.

## IV. Multi-rate Decoder Architecture

A new data flow graph, as shown in Fig. 4, is designed based on the TDMP and on the value-reuse property of OMS described above. By changing the parameter $k$ supplied to the CNU and varying the parameter $j$, the number of block rows to be processed, this architecture supports all the codes in the IEEE802.11n standard. Assuming a parallelization factor of $M$, the complete $P$ vector of size $z$ for one block column is available in $M$ memory banks of depth $s = \lceil \frac{z}{M} \rceil$, $P$ buffer is initialized to be channel values in the first sub-iteration. The shifter is constructed as $M \times M$ cyclic down logarithmic shifter, which is is composed of $\log_2(M)$ stages of $M$ multiplexors. Cyclic up shift by $u$ can be simply achieved by doing cyclic down shift of $z - u$. With one cyclic shifter, we can achieve $27 \times 27$, $54 \times 54$ or $81 \times 81$ shifting. This is illustrated by an example in Fig. 5. In Fig. 5(a), we have 8 messages stored in the $P$ buffer, if a cyclic up shift of 2 is needed, we only need to correctly generate the read addressing of P message and apply corresponding shifting on the logarithm shifter. For the case of parallelization of $M = 8$, as shown in 5(b), one layer of 2-to-1 multiplexors are needed, so that the 8 messages could be grouped into two sets and then corresponding shift is accomplished. At the same time, this arrangement could achieve $4 \times 4$ shifting. Similarly, base matrices with different expansion factors that are multiples of 27 can be supported using $27 \times 27$ shifter as the base module.

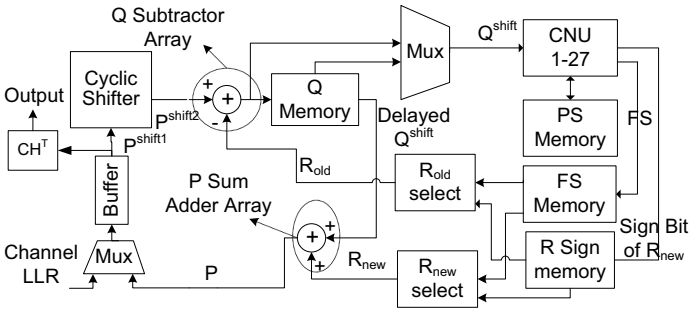


Fig. 4. Proposed LDPC decoder architecture.

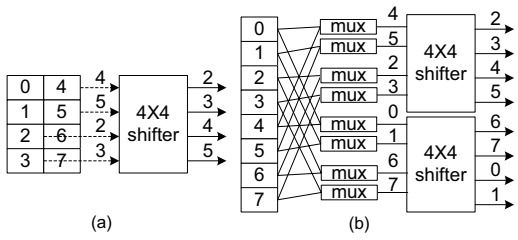


Fig. 5. (a) $8 \times 8$ shifting with $4 \times 4$ cyclic shifter for a parallelization $M = 4$. (b) $8 \times 8$ shifting with two $4 \times 4$ cyclic shifters for a parallelization $M = 8$.

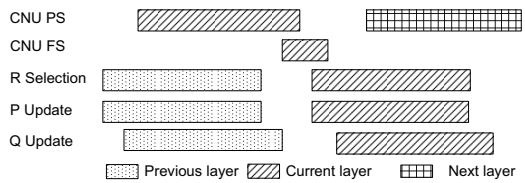


Fig. 6. Pipeline of proposed decoder.

The CNU array is composed of $M$ serial CNUs described in section III, the R selection units for $R_{new}$ ($R^i_{l,n}$ in (8)) are shown outside of the CNU units. Input vector to the CNU array can take values from either $Q$ memory bank or the newly updated $Q$ message from Q subtractor array. The $Q$ memory is partitioned into four banks dual access memory in order to avoid memory access conflict, which comes from the fact that the writing $\vec{Q}^i_{l-1,n}$, reading $\vec{Q}^i_{l-1,n-1}$ for CNU processing and reading $\vec{Q}^i_{l,n}$ for $P$ computation may happen for different locations in $Q$ memory. There is no control overhead for the partition when the lower two bits of address serve as bank selection. For some of the base matrices, re-ordering the layers is also done to avoid memory conflicts. The shift $S(l,n)$ imposed on $\vec{P}_n$ has to be undone and a new shift $S(l+1,n)$ imposed. This could be achieved by simply imposing a shift corresponding to the difference of $S(l+1,n)$ and $S(l,n)$. The CNU array then is ready to start the partial state computation for next block row as soon as one $Q$ message is produced. Final state of previous block rows, in which the compact information for $R$ messages is stored, are needed for TDMP as shown in (7), they are stored in the FS memory. There is $M$ memory banks of depth $j$ connected with each of the $M$ CNUs. Each of the FS memory banks has word length of 20 bits, constituted of {M1, -M1, +/-M2} with offset correction, and $M1$ index. In addition we need another $M$ memory banks with depth of $s$ to store the partial state with a word length of 16 bits as we need to store and retrieve (M1, M2, M1 index and cumulative sign). The sign bits of $R$ are stored in sign memory.

For the decoding of one layer, (7-9) are performed in sequence and these steps are repeated for all the layers. As shown in Fig. 6, the CNU array operates on the $R$ messages and partial states of two adjacent block rows (layers). While the final state has dependency on partial states, $P$ and $Q$ messages are dependent on the final states. In the decoding process, a block row of check nodes are processed in serial fashion, the output of the CNU is also in serial form. As soon as the output vector $\vec{R}^i_{l,n}$ corresponding to each block column $n$ in $H$ matrix for a block row $l$ is available, this could be used to produce updated sum $\left[\vec{P}_n\right]^{-S(l,n)}$ in (9), which is then immediately used in (7) to process the shifted vector for block row $l+1$. To accommodate the irregularity in block LDPC codes, the R selection unit for $R_{old}$ ($R^{i-1}_{l,n}$ in (7)) and PS processing are executed in linear order for the current layer (i.e. first non-zero block, second non-zero block in a layer), while order of R generation for $R_{new}$ processing is determined by the non-zero blocks of the next layer that has to be processed because $\vec{Q}^i_{l,n}$ in (7) has dependency on $\vec{R}^i_{l,n}$ in (8) of the previous layer. Furthermore, since check node degree of each layer in IEEE 802.11n varies widely from 7 to 21, it is not efficient that each layer executes for number of clock cycles equal to the maximum check-node degree. The R select unit for $R_{new}$ may be operating on any of the previous layers. It should be pointed out that R generation is independent of PS or FS processing, so its out-of-order processing will not impose any additional restriction on the architecture. Even though it does require careful scheduling and there will be some additional logic to account for selecting based on the M1 index.

## V. Implementation results and Discussion

The proposed TDMP architecture features large memory savings, up to $2x$ throughput advantage, as well as 50% less interconnection complexity. All the code lengths ($648, 1296$ and $1944$, according to different expansion factors $z = 27, 54$, and $81$ respectively) and code rates ($1/2, 2/3, 3/4$ and $5/6$) as specified in the IEEE 802.11n standard [13] are supported in this architecture. Table I gives the FPGA implementation and ASIC results for $M = 81$ are shown in Table II, in which VNU constitutes the $P$ adder array and $Q$ subtractor array. Note that the relatively large memory area in ASIC implementation is due to the 133 shallow memory banks required for the total number of $55344$ bits. Similar memory area overheads are

reported in [3]. Recent work on IEEE 802.11n LDPC decoder [15] consumes $375.14K$ logic gates and $88452$ bits of memory for $940$ Mbps throughput. Here, all calculations for the decoded throughput are based on an average of 5 decoding iteration to achieve frame error rate of $10^{-4}$, while $it_{max}$ is set to 15. The total power dissipation is estimated to be $238.4mW$ by the Synopsys design analyzer.

The memory efficiency comes from three-folded optimization. First, the TDMP facilitates removal of memory needed to store the channel LLR values as these values are implicitly stored in the $Q$ messages. The number of $Q$ messages that need to be stored are equal to $N_b \times z_{max} \times 5$, as opposed to storing $N_{nz} \times z_{max} \times 5$ messages in TPMP architectures where $N_b$ and $N_{nz}$ are the number of block column and total number of non-zero block columns in the $H$ matrix, respectively. Here $N_{nz} \approx 100$, $N_l = 12$ is the maximum number of layers or block rows by considering the base $H$ matrix, $k_{max} = 21$ is the maximum check-node degree, $N_b = 24$ and, $z_{max} = 81$ for block LDPC codes in IEEE 802.11n. The factor 5 comes from 5-bit quantization for $Q$ messages. Second, instead of storing all the $R$ messages, only the compressed information (final states in CNU) is stored in the proposed architecture. The total savings in R memory is $\frac{5k_{max}N_l z_{max} - [k_{max}+5\times3+[log_2(k_{max})]+1]N_l z_{max}}{5k_{max}N_l} \times 100\%$. The factor 5 comes from 5-bit quantization for $R$ messages. Furthermore, due to the block serial scheduling in the proposed architecture, there is only need to store $P$ messages for two blocks. The total savings of memory bits is $\frac{N_b \times z_{max} \times 6 - 2 \times z_{max} \times 6}{N_b \times z_{max} \times 6} \times 100\%$, where 6 is word-width of the $P$ messages. So the total savings in memory accounting for $R$ memory, $Q$ memory, and P memory, when compared to TPMP architectures based on SP [11] and min-sum [8], [9] is 63%. When compared to other TDMP architectures based on BCJR [3] and SP [5], the total memory savings is 55% since all the TDMP architectures have the same savings in $Q$ memory.

TABLE I
FPGA IMPLEMENTATION RESULTS FOR THE MULTI-RATE DECODER. FULLY COMPLIANT TO IEEE 802.11N (DEVICE, XILINX 2V8000FF152-5, FREQUENCY 110MHZ)

| | $M = 27$ | $M = 54$ | $M = 81$ | Available |
|---|---|---|---|---|
| Slices | 1836 | 3647 | 5514 | 46592 |
| LUT | 3317 | 6335 | 9352 | 93184 |
| SFF | 1780 | 3560 | 5341 | 93184 |
| BRAM | 46 | 89 | 133 | 168 |
| Memory(bits) | 56640 | 56640 | 55344 | |
| Throughput(Mbps) | | | | |
| $z = 81$ | 119 | 238 | 356 | |
| $z = 54$ | 119 | 238 | 178 | |
| $z = 27$ | 119 | 119 | 119 | |

TABLE II
ASIC IMPLEMENTATION RESULTS FOR THE MULTI-RATE DECODER FOR $M = 81$ (0.13 $\mu$M TECHNOLOGY [16], FREQUENCY 500MHZ)

| Resource | Area ($mm^2$) | Equivalent NAND-2 gates |
|---|---|---|
| CNUs | 0.45 | 67500 |
| VNUs | 0.07 | 10125 |
| Storage | 1.04 | N/A |
| Flip-flops | 0.03 | 3375 |
| Shifter and wiring | 0.22 | 18900 |
| total | 1.85 | 99900 |
| Throughput(Mbps) | 541, 1082 and 1618 for $z = 27, 54$ and $81$ | |

In terms of throughput and interconnect advantage, to achieve the same BER as that of TPMP schedule on OMS, TDMP schedule on OMS needs half the number of iterations. This essentially doubles the throughput. However, the choice of finite precision OMS results in a performance degradation of less than 0.1 dB 5-bit uniform quantization for $R$ and $Q$ messages and 6-bit uniform quantization for $P$ messages is used. The step size for quantization and offset parameter are set based on the code parameters. Moreover, due to the efficient data-flow graph, this architecture requires only one cyclic shifter, while the work in [3], [5], [7], [9] used two cyclic shifters.

## VI. CONCLUSION

We present a memory efficient multi-rate decoder architecture for turbo decoding message passing of block LDPC codes of IEEE 802.11n using the OMS algorithm for check-node update. Our work offers several advantages when compared to the other-state-of -the-art LDPC decoders in terms of significant reduction in logic, memory, and interconnect. This work retains the key advantages offered by the original TDMP work - however, our contribution is in using the value-reuse properties of OMS algorithm and devising a new TDMP decoder architecture to offer significant additional benefits.

## REFERENCES


[1] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes", *Electronics Letters*, volume 32, pp. 1645-1646, Aug 1996.

[2] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier and X. Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Communications*, vol. 53, pp. 1288-1299, Aug. 2005.

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

[4] H. Sankar and K. Narayanan, "Memory-efficient sum-product decoding of LDPC codes," *IEEE Trans. on Communications*, vol.52, pp. 1225-1230, Aug. 2004.

[5] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pp. 107- 112, Oct. 2004.

[6] A. Blanksby, C. 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, Iss.3, pp. 404-412, Mar 2002.

[7] K. Gunnam, G. Choi and M. Yeary, "An LDPC decoding schedule for memory access reduction," *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 173-6 vol. 5, May 2004.

[8] M. Karkooti and J. Cavallaro, "Semi-parallel reconfigurable architectures for real-time LDPC decoding," *Proceedings of International Conference on Information Technology, Coding and Computing*, vol. 1, pp. 579-585, 2004.

[9] T. Brack, F. Kienle and N. Wehn. "Disclosing the LDPC code decoder design space", *Design Automation and Test in Europe (DATE) Conference*, pp. 200-205, March 2006.

[10] K. Gunnam, W. Wang, E. Kim, G. Choi and M. B. Yeary, "Decoding of Array LDPC Codes using On-The-Fly Computation," Accepted for *40th Asilomar Conf. on Signals, Systems and Computers*, October 2006.

[11] H. Zhong, T. Zhang, "Block-LDPC: a practical LDPC coding system design approach," *IEEE Trans. on Circuits and Systems I*, vol.52, no.4, pp. 766- 775, April 2005.

[12] IEEE 802.16e WiMax Standard, *IEEE P802.16e-2005*, October 2005.

[13] *IEEE 802.11 Wireless LANsWWiSE Proposal: High Throughput extension to the 802.11 Standard.* IEEE 11-04-0886-00-000n.

[14] K. Gunnam and G. Choi, "A Low Power Architecture for Min-Sum Decoding of LDPC Codes," *TAMU, ECE Technical Report*, May, 2006, TAMU-ECE-2006-02. [Online]. Available: http://dropzone.tamu.edu/tehpubs.

[15] M. Rovini, N. L'Insalata, F. Rossi, L. Fanucci, "VLSI design of a high-throughput multi-rate decoder for structured LDPC codes," in *Proc. 8th Euromicro Conference on Digital System Design*, pp. 202-209, Sept. 2005

[16] Open source standard cell library. Available online: http://www.vlsitechnology.org