This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax

Kiran K. Gunnam[1], Gwan S. Choi[1], Mark B. Yeary[2] and Mohammed Atiquzzaman[3]
[1]Department of ECE, Texas A&M University, College Station, TX-77843
[2]School of Electrical and Computer Engineering, University of Oklahoma, Norman, OK-73109
[3]School of Computer Science, University of Oklahoma, Norman, OK-73109

Abstract— We present a new multi-rate architecture for decoding irregular LDPC codes in IEEE 802.16e WiMax standard. The proposed architecture utilizes the value–reuse property of offset min-sum, block-serial scheduling of computations and turbo decoding message passing algorithm. The decoder has the following advantages: 55% savings in memory, reduction of routers by 50%, and increase of throughput by 2x when compared to the recent state-of-the-art decoder architectures.

***Index Terms*— low-density parity-check (LDPC) codes, offset min-sum, on-the-fly computation, decoder architecture, layered decoding, turbo-decoding message passing, irregular LDPC,IEEE 802.16e.**

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 the decoding algorithm 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 a two phase schedule. Various algorithms are available for check-node updates and widely used algorithms are sum of products (SP), min-sum (MS), and Jacobian-based BCJR (named after its discoverers Bahl, Cocke, Jelinik, and Raviv). The authors in [2] introduced the concept of turbo decoding message passing (TDMP, sometimes also called layered decoding) using BCJR for their architecture-aware LDPC (AA-LDPC) codes. TDMP offers 2x throughput and significant memory advantages when compared to TPMP. TDMP is later studied and applied for different LDPC codes using sum of products algorithm and its variations in [3]-[4]. TDMP is able to reduce the number of iterations required by up to 50% without performance degradation when compared to the standard message passing algorithm. A quantitative performance comparison for different check updates was given by Chen and Fossorier et al. [5]. Their research showed that the offset min-sum (OMS) decoding algorithm with 5-bit quantization could achieve the same bit-error rate (BER) performance as that of floating point SP and BCJR with less than 0.1 dB penalty in SNR.

While fully-parallel LDPC decoder designs [6] suffered from complex interconnect issues, various semi-parallel implementations based on structured LDPC codes [2],[7]-[9],[13]-[14] alleviate the interconnect complexity. All the structured LDPC codes share the property that the H matrix is constructed out of cyclic shifted version of identity matrix and null matrices. In this work, we propose to apply TDMP for the offset MS for block LDPC codes used in IEEE 802.16e (Mobile WiMax). WiMax technology involves microwaves for the transfer of data wirelessly. It can be used for high-speed, mobile wireless networking at distances up to a few miles. The main contribution of this work is an efficient architecture, that utilizes the value–reuse property of OMS, cyclic shift property of structured LDPC codes and enhancement of our previous work of block serial scheduling [7]. The resulting decoder architecture has, to our best knowledge, the lowest requirements of logic, interconnection, and memory.

The rest of the paper is organized as follows. Section II introduces structured block LDPC codes, OMS 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 offset MS is shown in Section IV. Section V presents the FPGA implementation results and discussion. Section VI concludes the paper.

II. LDPC CODES AND DECODING

A. Block LDPC Codes of WiMax

The block irregular LDPC codes have competitive performance and provide flexibility and low encoding/decoding complexity [10]. The entire H matrix is composed of the same style of blocks with different cyclic shifts, which allows structured decoding and reduces decoder implementation complexity. Each base H matrix in block LDPC codes has 24 columns, simplifying the implementation. Having the same number of columns between code rates minimizes the number of different expansion factors that have to be supported. There are four rates supported: 1/2,2/3,3/4, and 5/6, and the base H matrix for these code rates are

1-4244-0353-7/07/$25.00 ©2007 IEEE

defined by systematic fundamental LDPC code of M_b-by-N_b where M_b is the number of rows in the base matrix and N_b is the number of columns in the base matrix. The following base matrices are specified: 12 x 24, 8 x 24, 6 x 24, and 4 x 24. The base model matrix is defined for the largest code length (N = 2304) of each code rate. The set of shifts in the base model matrix are used to determine the shift sizes for all other code lengths of the same code rate. Each base model matrix has 24 (= N_b) block columns and M_b block rows. The expansion factor z is equal to N/24 for code length N. The expansion factor varies from 24 to 96 in the increments of 4, yielding codes of different length. For instance, the code with length N = 2304 has the expansion factor z=96 [10]. Thus, each LDPC code in the set of WiMax LDPC codes is defined by a matrix H as

$$H = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,N_b} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,N_b} \\ \cdots & \cdots & \cdots & \cdots \\ P_{M_b,1} & P_{M_b,2} & \cdots & P_{M_b,N_b} \end{bmatrix} = P^{H_b} \quad (1)$$

where $P_{i,j}$ is one of a set of *z-by-z* cyclically right shifted identity matrices or a *z-by-z* zero matrix. Each 1 in the base matrix H_b is replaced by a permuted identity matrix while each 0 in H_b is replaced by a negative value to denote a *z-by-z* zero matrix.

B. Offset min-sum decoding algorithm

Assume binary phase shift keying (BPSK) modulation (a 1 is mapped to -1 and a 0 is mapped to 1) over an additive white Gaussian noise (AWGN) channel. The received values y_n are Gaussian with mean $x_n = \pm 1$ and variance σ^2. The reliability messages used in belief propagation (BP)-based offset min-sum algorithm can be computed in two phases: 1. check-node processing and 2. variable-node processing. The two operations are repeated iteratively until the decoding criterion is satisfied. This is also referred to as standard message passing or two-phase message passing (TPMP). For the i^{th} iteration, $Q_{nm}^{(i)}$ is the message from variable node n to check node m, $R_{mn}^{(i)}$ is the message from check node m to variable node n, $\mathrm{M}(n)$ is the set of the neighboring check nodes for variable node n, and $\mathrm{N}(m)$ is the set of the neighboring variable nodes for check node m. The message passing for TPMP based on OMS is described in the following three steps as given in [11] to facilitate the discussion on TDMP in the next section:

Step 1. *Check-node processing*: for each m and $n \in \mathrm{N}(m)$,

$$R_{mn}^{(i)} = \delta_{mn}^{(i)} \max\left(\kappa_{mn}^{(i)} - \beta, 0\right), \quad (2)$$

$$\kappa_{mn}^{(i)} = \left|R_{mn}^{(i)}\right| = \min_{n' \in \mathrm{N}(m) \setminus n} \left|Q_{n'm}^{(i-1)}\right|, \quad (3)$$

where β is a positive constant and depends on the code parameters [5]. In general, for the irregular codes, we will also apply the correction on variable node messages. The sign of check-node message $R_{mn}^{(i)}$ is defined as

$$\delta_{mn}^{(i)} = \left(\prod_{n' \in \mathrm{N}(m) \setminus n} \mathrm{sgn}\left(Q_{n'm}^{(i-1)}\right) \right),$$

Step 2. *Variable-node processing*: for each n and $m \in \mathrm{N}(n)$,

$$Q_{nm}^{(i)} = L_n^{(0)} + \sum_{m' \in \mathrm{M}(m) \setminus m} R_{m'n}^{(i)}, \quad (4)$$

where the log-likelihood ratio of bit n is $L_n^{(0)} = y_n$.

Step 3. *Decision: for final decoding*

$$P_n = L_n^{(0)} + \sum_{m \in M(n)} R_{mn}^{(i)}. \quad (5)$$

A hard decision is taken by setting $\hat{x}_n = 0$ if $P_n(x_n) \geq 0$, and $\hat{x}_n = 1$ if $P_n(x_n) < 0$. If $\hat{x}H^T = 0$, the decoding process is finished with $\hat{x}_n$ as the decoder output; otherwise, repeat steps (1-3). If the decoding process doesn't end within predefined maximum number of iterations, $it_{\max}$, stop and output an error message flag and proceed to the decoding of the next data frame.

In TDMP, the block LDPC with j block rows can be viewed as concatenation of j layers or constituent sub-codes similar to observations made for AA-LDPC codes in [2]. In TDMP, after the check-node processing is finished for one block row, the messages are immediately used to update the variable nodes (2), whose results are then provided for processing the next block row of check nodes (1). 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, variable messages are initialized as channel values and are used to process the check nodes of the first block row. After completion of that block row, variable messages are updated with the new check- node messages. This concludes the first sub-iteration. In similar fashion, the result of check-node processing of the second block row is immediately used in the same iteration to update the variable-node messages for third block row. The completion of check-node processing and associated variable-node processing of all block rows constitutes one iteration.

The TDMP can be described with (6-9):

$\vec{R}_{l,n}^{(0)} = 0, \vec{P}_n = \vec{L}_n^{(0)}$ [Initialization for each new received data frame], (6)

$\forall i = 1,2,\cdots, it_{\max}$, [Iteration loop]

$\forall l = 1,2,\cdots, j$, [Sub-iteration loop]

$\forall n = 1,2,\cdots,k$, [Block column loop]

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

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

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

where the vectors $\vec{R}_{l,n}^{(i)}$ and $\vec{Q}_{l,n}^{(i)}$ represent all the R and Q messages in each non-zero block of H matrix, $s(l,n)$ denotes the shift coefficient for the l^{th} block row and n^{th} non-zero block of the H matrix (note that null blocks in the H matrix need not be processed); $\left[\vec{R}_{l,n}^{i-1}\right]^{S(l,n)}$ denotes that the vector $\vec{R}_{l,n}^{i-1}$ is cyclically shifted up by the amount $s(l,n)$, k is the check-node degree of the block row. A negative sign on $s(l,n)$ indicates that it is cyclic down shift (equivalent cyclic left shift). $f(\cdot)$ denotes the check-node processing, which can be done using BCJR, SP or MS. For the proposed work we use MS as defined in (1-3).

III. VALUE-REUSE PROPERTIES OF CHECK-NODE PROCESSING

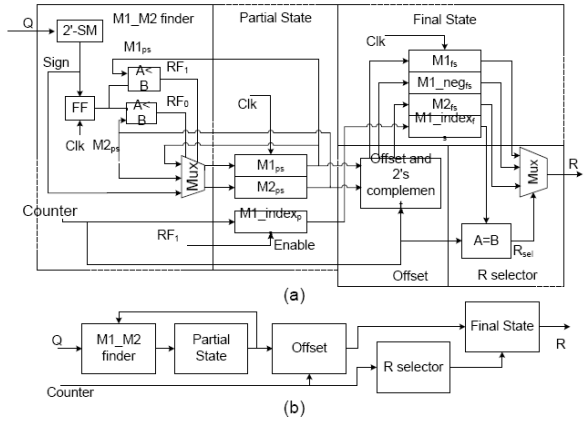

Fig 1. Serial CNU for OMS using value-reuse property.

This section presents the micro-architecture of serial CNU for OMS, which was used in our recent work on TPMP architecture [11]-[12]. The same CNU can be used in TDMP architecture presented in the next section. For each check node m , $\left|R_{mn}^{(i)}\right|$ $\forall n \in \mathrm{N}(m)$ takes only 2 values. The least minimum and the second least minimum of the entire set of the messages can be defined from various variable-nodes to the check-node m as,

$$M1_m^{(i)} = \min_{n' \in \mathrm{N}(m)} \left|Q_{mn'}^{(i-1)}\right|., \quad (10)$$

$$M2_m^{(i)} = 2nd\min_{n' \in \mathrm{N}(m)} \left|Q_{mn'}^{(i-1)}\right|. \quad (11)$$

Now (3) becomes

$$\left|R_{mn}^{(i)}\right| = M1_m^{(i)}, \ \forall n \in \mathrm{N}(m) \setminus M1\_index \quad (13)$$

$$= M2_m^{(i)}, n = M1\_index .$$

Since $\forall n \in \mathrm{N}(m)$, $\delta_{mn}^{(i)}$ takes a value of either $+1$ or -1 and $\left|R_{mn}^{(i)}\right|$ takes only two values. Equation (2) gives rise to only three possible values for the whole set $R_{mn}^{(i)}$ $\forall n \in \mathrm{N}(m)$. In a VLSI implementation, this property greatly simplifies the logic and reduces the memory. Since only the two least minimum numbers need to be identified, the number of comparisons can be reduced, while in the CNU proposed in [8], *2k* comparators are used to compute (3) associated with that check node. We present efficient serial implementations for CNU. Fig. 1(a) shows the CNU micro-architecture for (3, 19) code. In the first *19* clock cycles of the check-node processing, incoming variable messages are compared with the two up-to-date least minimum numbers (partial state, PS) to generate the new partial state, *M1*, which is the least minimum value, *M2*, which is the second minimum value and index of *M1*. The final state (FS) is then computed by offsetting the partial state. It should be noted that the final state includes only $M1_m^{(i)}$, $-M1_m^{(i)}, \pm M2_m^{(i)}$ with offset correction. Figure 1(b) is the block diagram of the same architecture. M1_M2 finder computes the two least numbers and stores them in partial state. The offset module applies the offset correction, and stores the results in the final state module. R Selector then selects the output R messages. In operation, the final state and partial state will operate on different check nodes simultaneously. Normally CNU (check-node unit) processing is done using the signed magnitude arithmetic for (1-2) and VNU (variable-node unit processing) (4-5) is done in 2's complement arithmetic. This requires 2's complement to signed conversion at the inputs of CNU and signed to 2's complement at the output of CNU. In the proposed scheme, 2's complement is applied to only 2 values instead of k values at the output of CNU. The value re-use property also reduces the memory requirement significantly. Conventionally, the number of messages each CNU stores is equal to the number of edges it has, that is k . Now only four units of information are needed: the three values that $R_{mn}^{(i)}$ may take and the location of $M1_m^{(i)}$ The memory savings due to this value re-use property of OMS [11]-[12] for WiMax LDPC codes is quantified in section V.

IV MULTI-RATE DECODER ARCHITECTURE USING TDMP AND OMS

A. Architecture Description

A new data flow graph is designed based on the TDMP and on the value-reuse property of the OMS algorithm described above(see Fig. 2). For ease of discussion, we will illustrate the architecture for the specific structured code denoted as rate ¾ code A. Note that all the codes have the same number of block columns. By changing the parameter k supplied to the CNU and by varying the parameter *j*, the number of block rows to be processed, this architecture supports all the codes in the 802.16e standard. For rate 3/4 code A of length 1152 has j =6

block rows and check-node degree of the 6 block rows is given by k_v= [13 12 12 12 12 13] and the block size is z = 48. Assume that the desired parallelization is M=24. The cyclic shifter needed is $M \times M$. The $z \times z$ cyclic shift is achieved with cyclic shifts of $M \times M$ in combination with the appropriate address generation and this works for only z=24,48 and 96. The complete P vector of size z is available in M memory banks of depth $s = ceil(z/M) = 2$. The shifter is constructed as a cyclic down logarithmic shifter to achieve the cyclic shifts specified by the binary encoded value of the shift. The logarithmic shifter is composed of $\log 2(M)$ stages of M 2-in-1 multiplexers. Cyclic up shift by u can be simply achieved by doing cyclic down shift with $z-u$ on the vector of size. Say now if we want to change the parallelization M to 48. If we construct a single 48 x 48 cyclic shifter, it can only handle z=48. So, we use two 24 x 24 cyclic logarithmic shifters to construct the 48 x 48 shifter while being able to work as two independent 24 x24 shifters to support the expansion factor z=24. We need to introduce some additional multiplexers to achieve this. This way the decoder can support the expansion factors of 24, 48 and 96. Similarly, the cyclic shifter implementation for M=96, is constructed out of 4 24 x 24 cyclic logarithmic shifters. One should note that it is not possible to achieve cyclic shifts specified by $s(l,n)$,(=0,1,..z-1)on a vector of length z with a cyclic shifter of size $M \times M$ if M is not a integer multiple of z.

So to be able to accommodate different shifts needed, we can use a Benes network as in [15], which is of complexity $2\log 2(M)-1$ stages of M 2-in-1 multiplexers. A memory can be used to store control inputs needed for different shifts in case of supporting one expansion factor [2],[15].[2] uses Omega network, which is less complex than Benes network[15]. However both [12] and [15] will support only base H matrix. Note that this memory for providing control signals to this network is equal to $\frac{M}{2}(2\log 2(M)-1)$ bits for every shift value that needs to be supported. This will be a very huge requirement for supporting all the WiMax codes. Note that, the memory needed for storing control signals for Omega network is around 1.22 mm^2 in out of the decoder chip area of 14.1 mm^2.[2]. This is equivalent to storing the control signals for one expansion factor and one base H matrix. So if the same kind of scheme is used to support 19 different expansion factors and 6 types of base H matrices *in run time*, the control signal memory needs approximately 139.08 mm^2. So this approach clearly will not work. We propose a simpler approach to generate the control signals using a Master-Slave Benes router (Fig. 4). Assume that we need to perform a cyclic shift of 2 on a message vector of length 4 using a 8 x 8 Slave Benes network. Supply the integers(2,3,0,1,4,5,6,7) to the Master Benes network which is always configured to sort the inputs and output (0,1,2,…7). During the sorting process, the Master Benes network can generate the control signals on by virtue of comparators. These signals can be used in the Master network to accomplish sorting. Also these signals can be used in the Slave network to achieve the desired shift of 2 Note that the complexity of this approach adds additional logic requirements of a decoder that is optimized for supporting one or limited number of base H matrices, i.e., when we replace the logarithmic cyclic shifter with the Master-Slave Benes cyclic shifter. For more implementation details, please refer to [16-17].

B. Decoder Operation

All the check-node processing and variable-node processing is done in a time division multiplexed fashion for each sub-vector of length as shown in Fig. 3. To process a block in a block row (layer), it takes s clock cycles. A check-node process unit (CNU) is the serial CNU based on OMS described in the previous section. The CNU array is composed of M serial CNUs described in section 3. As shown in the pipeline (Fig. 3), the CNU array operates on the R messages and partial states of two adjacent block rows. While the final state has dependency on partial states, P and Q messages are dependent on the final states. Since final state of previous block rows, in which the compact information for CNU messages is stored, is needed for TDMP, it is stored in the FS memory. There is one memory bank of depth j , which is 12 in this case, connected with each CNU. The FS memory for the entire CNU array is implemented as M banks of memory with depth js and word length 20 bits, constituted of {M1, -M1, +/-M2} with offset correction, and M1 index. In addition, we need another memory with M banks with depth equal to s to store the partial state, with the word length 16 bits as we need to store and retrieve (M1, M2, M1 index and cumulative sign). Note that we need to store partial state for only one block row at any time. In the decoding process, a block row of check nodes are processed in serial fashion using M CNUs as in (8), the output of the CNU is also in serial form. The CNU array will start the partial state computation for next block row as soon as the partial state processing for the previous block row is done. The Q messages that are fed in the present block row/layer are dependant on R messages of the current layer as well as the R messages belonging to the different blocks of different layers. perform the R selection. This can be accomplished with out-of-order processing of R_{new} message generation. An R select unit generates the R messages for k edges of a check node from three possible values stored in final state memory word associated with that particular check node in a serial fashion. Its functionality and structure is the same as the block denoted as R select in CNU. This unit can be treated as a de-compressor of the check-node edge information, which is stored in compact form in FS memory.It is possible to do the decoding using a different sequence of layers instead of processing the layers from 1 to j which is typically used to increase the parallelism such that it is possible to process two block rows simultaneously [4]. In this work, we use the concept of re-ordering of layers for increased parallelism as well as for low complexity memory implementation and also for inserting additional pipeline stages without incurring overhead.

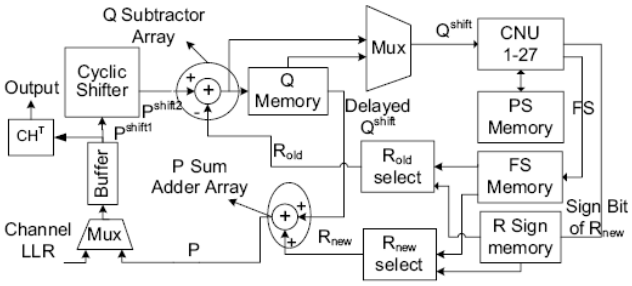

Fig. 2: Multi-rate LDPC decoder architecture for Block LDPC codes

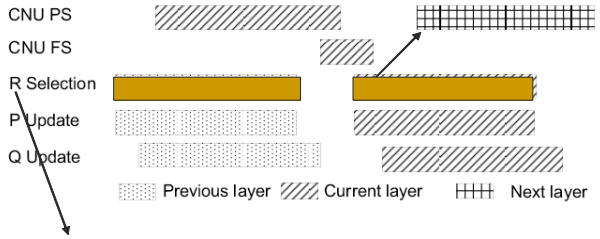

Fig. 3. Pipeline for the layered decoding for irregular QC-LDPC codes(Block LDPC codes of IEEE 802.16e)

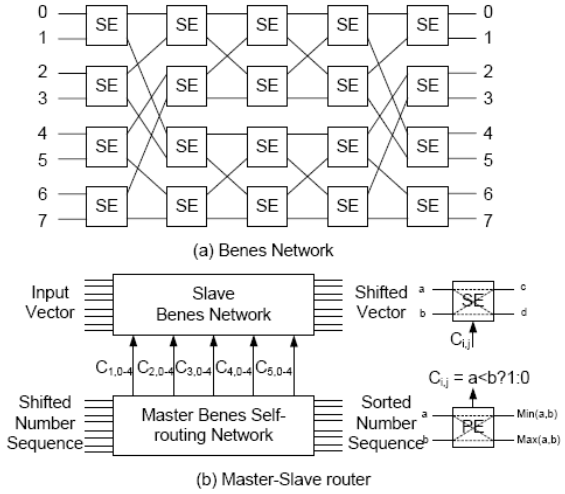

Fig.4. Proposed Master-slave router to support different cyclic shifts that arise due to the a wide range of expansion factors z(=24,28,..,96) and shift coefficients(0,1,..,z-1).

V Discussion and FPGA Implementation Results

Table I gives the FPGA implementation results. The proposed TDMP architecture features large memory savings, up to 2x throughput advantage, as well as 50% less interconnection complexity. The TDMP permits us to use a running sum, which is initialized to channel log-likelihood ratio (LLR) values in the first iteration. *So there is no memory needed to store the channel LLR values as these values are implicitly stored in the Q messages.* Since the maximum number of Q messages that need to be stored are equal to $N_b \times z_o \times 5$, as opposed to storing $N_{nz} \times z_o \times 5$ messages in TPMP architectures where N_b is the maximum number of block columns of all the codes that need to be supported, z_o is the maximum expansion factor of the base matrix, N_{nz} is the number of non-zero blocks by considering the base H matrix, which has the maximum number of non-zero blocks among all the base H matrices that need to be supported. For WiMax LDPC codes, these parameters are $N_b = 24$, $z_o = 96$, and $N_{nz} = 76$. So, the total savings in Q memory are 68% as a direct result of employing TDMP proposed in [2].

Instead of storing all the R messages, the compressed information cumulative sign, *M1*, *-M1*, *+/-M2,* and index of *M1* is stored. R select unit can generate the R message by the use of an index comparator and the XOR of the cumulative sign and the sign bit of the corresponding Q message which comes from the sign FIFO. The total savings in R memory is $\frac{5kN_l z_o - [k+5\times 3+ceil(\log_2(k))+1]N_l z_o}{5kN_l}\times 100\%$, where N_l is the number of layers or block rows by considering the base H matrix, which has the maximum number of non-zero blocks among all the base H matrices that need to be supported. The factor 5 comes due to the use of 5-bit quantization for R messages. Among the different base LDPC codes in WiMax, rate 5/6 code has the maximum check-node degree, $k = 19$ and the maximum number of block rows in the *H* matrix is 12. So the savings of R memory is 57%.

Also note that, due to the nature of block serial scheduling and the scheduling of layered processing in the architecture, there is only need to store the P messages for only two blocks. The total savings of memory bits is $(N_b \times z_o \times 6 - 2\times z_o \times 6)/(N_b \times z_o \times 6)\times 100\%$. Note that the factor 6 comes due to the number of bits used to represent the P message. So the savings are around 91% as $k_{\max} = 19$ and, z_0=*96* for block LDPC codes in 802.16e.

The total savings in memory accounting for R memory, Q memory, and P memory, when compared to TPMP architectures based on SP [13] and min-sum [7], [8],[14] is 63%.When compared to TDMP architecture based on BCJR [2], the total memory savings is 55% since both architectures have the same savings in Q memory.

TABLE I

FPGA IMPLEMENTATION RESULTS OF THE MULTI-RATE DECODER (supports z=24,48 and 96 and all the code rates) (Device, *Xilinx 2V8000ff152-5,* frequency *110MHz)*

	Used			Available
	M=24	M=48	M=96	
Slices	1640	3239	6568	46592
LUT	2982	5664	11028	93184
SFF	1582	3165	6330	93184
BRAM	38	73	100	168
Memory (bits)	65760	65760	60288	
Through-put (Mbps)	41~70	57~139	61~278	

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 when compared to the TDMP architecture. Moreover, this architecture requires only one cyclic shifter instead of two cyclic shifters [2], [4]. Note that the architecture features a partial state memory when compared to other architectures. However, this is small as it must contain the partial state for only one block row at any time, is equal to 1536 bits. In the case of parallelization equal to $M = z_0$, then there is no need for P buffer. Also, FS memory bank need to store only R messages belonging to 11 layers. The P buffer is not needed as the shifter employed is $z_o \times z_o$ and it can perform the shift without the need of a buffer since the input messages are available in the chunks of z_0. There is no need for PS memory bank, since there are z_0 CNU to handle the maximum number of rows in a block row (z_0), and consequently there is no time folding. The data throughput results are presented in Fig. 5. The implementation has a performance penalty of less than 0.15 dB in SNR when compared to floating point TDMP decoding. User data throughput t_u is given by $t_u = rate \times t_d$, where t_d is decoded throughput and is given by $t_d = Nf/(it_{\max} CCI)$, where N is the number of iterations, f is the decoder chip frequency and *CCI* stands for number of clock cycles required to complete one iteration. *CCI* is given by $CCI = N_{nz}\left\lceil \frac{z}{M} \right\rceil + 2N_b$. *The distinction of this architecture is that a near optimal minimal number of clock cycles are achieved when the expansion factor is a multiple of parallelization of the decoder*- Note that some codes support processing of two layers in parallel and the decoder can accommodate this if sufficient parallelism is available as can be seen from Fig. 5. It is also possible to exploit the parallelism completely for other expansion factors also when more than one frame can be processed simultaneously- however this requires additional buffering, so this scheme is not incorporated in the present design.

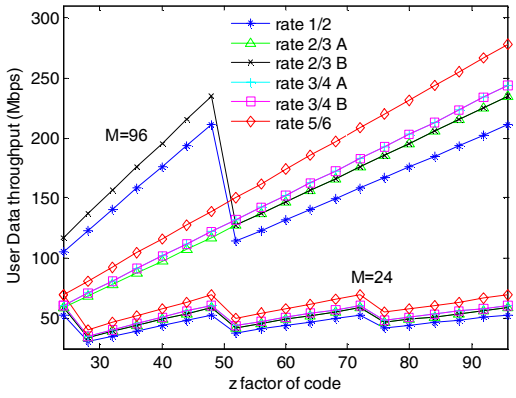

Fig. 5. User data throughput of the proposed decoder vs. the expansion factor of the code,z, for different numbers of decoder parallelization, M..

VI CONCLUSION

We present a memory efficient multi-rate decoder architecture for turbo decoding message passing of block LDPC codes of IEEE 802.16e 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 offset MS algorithm and devising a new TDMP decoder architecture to offer significant additional benefits.

VII REFERENCES

[1] D.J.C. MacKay and R.M. Neal. "Near Shannon Limit Performance of Low Density Parity Check codes" *Electronics Letters*, volume 32, pages 1645-1646, Aug 1996.

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

[3] H. Sankar and K.R. Narayanan, "Memory-efficient sum-product decoding of LDPC codes," *Communications, IEEE Transactions on*, vol.52, no.8pp. 1225- 1230, Aug. 2004

[4] Hocevar, D.E., "A reduced complexity decoder architecture via layered decoding of LDPC codes," *Signal Processing Systems*, 2004. SIPS 2004. IEEE Workshop on , .pp. 107- 112, 13-15 Oct. 2004

[5] 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.

[6] Blanksby, A.J.; Howland ,C.J, 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, Mar 2002 Pages:404-412

[7] K. Gunnam, G. Choi and M. B. 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] "*Part 16: air interface for fixed and mobile broadband wireless access systems amendment for physical and medium access control layers for combined fixed and mobile operation in licensed bands*", IEEE P802.16e-2005, October 2005

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

[12] 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.

[13] L. Yang; M. Shen; H. Liu, and C. Shi, "An FPGA implementation of low-density parity-check code decoder with multi-rate capability," *Proceedings of the Asia and South Pacific Design Automation Conference*, .pp. 760- 763 Vol. 2, 18-21 Jan. 2005

[14] H. Zhong and 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

[15] G. Malema and M. Liebelt, "Interconnection Network for Structured Low-Density Parity-Check Decoders," Asia-Pacific Conference on ,Communications, vol., no.pp. 537- 540, 03-05 Oct. 2005

[16] K. Gunnam, G. Choi, M. B. Yeary and M.Atiquzzaman, "VLSI architectures for layered decoding for irregular LDPC codes of WiMax", TAMU, ECE Technical Report, July 2006, TAMU-ECE-2006-08.

[17] K. Gunnam, "Area and energy efficient VLSI architectures for low-density parity-check decoders using an on-the-fly computation," PhD Dissertation, Texas A&M University, October 2006.