AN LDPC DECODING SCHEDULE FOR MEMORY ACCESS REDUCTION

Kiran Gunnam, Gwan Choi, Mark Yeary[*]

Dept. of Electrical Engineering, Texas A&M University, College Station, TX-77840
[*]ECE Dept., University of Oklahoma, Norman, OK-73109

ABSTRACT

Recent research efforts based on joint code-decoder design methodology have shown that it is possible to construct structured LDPC (Low Density Parity Check) codes without any performance degradation. An interesting new data independence property between the two classes of messages viz. check to bit and bit to check involved in decoding, is observed. This property is a result of the specific structuring of parity check matrix. By exploiting this property, we propose an architecture in which the computation of messages is synchronized such that each class of messages is consumed immediately by the computational unit for another class of messages. The internal memory of the check to bit units is increased in tune with the storage requirement of the check to bit messages. The separate memories for check to bit and bit to check messages are eliminated. This approach has memory savings of 75% and reduces the overall memory accesses by 66%.

1. INTRODUCTION

Low-Density Parity Check (LDPC) codes and Turbo codes are among the best known near Shannon limit codes [1]. LDPC decoding algorithm has more parallelization when compared to the decoding algorithm of Turbo codes. The major issues surrounding the VLSI implementation of LDPC decoders are the complex interconnects and large memory requirements due to the sparse nature of the parity generator matrix [2-3]. This paper proposes low complexity architecture with reduced memory requirements for LDPC decoding based on the recent work on structured LDPC codes [4-7].

LDPC codes can be described by an $m \times n$ parity check matrix H in which the average number of non-zero elements (i.e. one in GF2) in each row is a constant. In a (r,c) regular code, each of the n bit nodes $(b_1, b_{2,}..., b_n)$ has connections to r check nodes and each of the m check nodes $(c_1, c_{2,}..., c_m)$ has connections to c bit nodes. LDPC codes can be decoded by the Gallager's iterative belief-propagation (BP) algorithm [1].

The rest of the paper is organized as follows. Section 2 explains the code construction used and the relevant decoding schedule property which is exploited in the new architecture. Section 3 presents architecture to eliminate the message storage and Section 4 presents comparison with the existing work.

2. DECODING SCHEDULE

2.1. Code Design

We focus on the construction which structures the parity check matrix H into blocks of $p \times p$ matrices such that: 1. a bit in a block participates in only one check equation in the block and 2. each check equation in the block involves only one bit from the block. We show that this specific construction for codes having short block length (~2K bits) has some interesting properties which are not exploited in the previous art.

One method to perform this construction is through cyclotomic cosets [7]. Another method is to achieve this property by employing random bit filling algorithm (for low rate codes such as rate ½ codes) and geometric constructions (for high rate codes such as rate 8/9 codes) [5, 6]. The work [7] reports no performance degradation for a (3, 5) - LDPC code of length 1055, rate 0.4; constructed from cyclotomic cossets. The work [5] reports a minimal performance degradation up to 0.2 dB for a (3, 6) - LDPC code of length 2040, rate 0.5; constructed with random bit filling algorithm and when the "parallelization factor" [5] is set to p=2040/6=340 (the case of our interest). The work [6] reports a minimal performance degradation up to 0.2 dB for a (3, 30) - LDPC code of length 2070, rate 0.9; constructed with geometric methods and when "parallelization factor" [5] is set to p=2070/30=69 (the case of our interest).

The H matrix can be constructed with filling with matrices obtained by permuting identity matrix by the appropriate shift coefficients [7]. Say $B_{j,k}\ \forall j = 1,2..r; k = 1,2,..c$ is a $p \times p$ matrix, located at the j^{th} block row and k^{th} block column of H matrix. The scalar value $s(j,k)$ denotes the shift applied to $I_{p\times p}$ identity matrix to obtain the $(j,k)^{th}$ block, $B_{j,k}$, and

the rows in the $I_{p\times p}$ identity matrix are cyclically shifted to the right $s(j,k)$ positions for $s(j,k)\in\{0,1,2,...,p-1\}$.

Let us define S as a $c\times r$ shift coefficient matrix in which $S_{j,k}=s(j,k)\ \forall j=1,2..r; k=1,2,..c$. (1)

So an H matrix, in this construction, can be completely characterized by these two simple matrices viz. $I_{p\times p}$ and $S_{c\times r}$. To define H matrix, we start with fixing c, r and finding an appropriate p and shift coefficient matrix S such that the BER performance is maintained when compared to a random construction.

For example if $c=5, r=3$ and $p=211$ the use of cyclotomic cosets [7] results in the following shift coefficient matrix for the code of length $1055(n=cp)$.

$$S_{3\times5}=\begin{bmatrix}2 & 3 & 110 & 142 & 165\\ 5 & 64 & 96 & 113 & 144\\ 7 & 50 & 75 & 116 & 174\end{bmatrix} \quad (2)$$

The resulting H matrix has same BER performance when compared to a similar code with random construction [7].

2.2. Block Message Independence Property

The reliability messages used in Gallager's Belief Propagation algorithm can be computed in two phases viz. check node processing (3) and bit node processing (4) and this is repeated iteratively till the decoding criterion is satisfied [1,3]. The message passing equations are given by

$$R_{cj,bi}=\psi^{-1}\left[\left(\sum_{i'=Row[cj][1]}^{Row[cj][c]}\psi\left(Q_{i',cj}\right)\right)-\psi\left(Q_{bi,cj}\right)\right].\delta(cj,bi) \quad (3)$$

$$Q_{bi,cj}=\left(\sum_{j'=Col[bi][1]}^{Col[bi][r]}R_{j',bi}\right)-R_{cj,bi}+\wedge(bi) \quad (4)$$

where

$R_{cj,bi}$ is the message from check c_j to bit b_i, $Q_{bi,cj}$ is the message from bit b_i to check c_j, $\psi(x)=-\log\left(\tanh\left(x/2\right)\right)$ is the Gallager's function which is invariant under its inverse, $\delta(cj,bi)$ is ±1 and is given by

$$\delta(cj,bi)=\left(\operatorname{sgn}\left(Q_{bi,cj}\right).\prod_{i'\in Row[cj]}\operatorname{sgn}\left(Q_{i',cj}\right)\right).(-1)^{|Row[cj]|} \quad (5)$$

$(-1)^{|Row[cj]|}=1$ for codes constructed with even parity. $\wedge(bi)$ is the intrinsic reliability metric of bit i. $Row[c_j][1...c]$ ($Col[b_i][1...r]$) gives the locations of bits (checks) connected to the check node c_j (bit node b_i).

We can represent R and Q messages by the following matrices for deriving the new data independence property. This arrangement is similar to physical message storage employed in [3] except that these matrices are not really stored in the proposed architecture.

$$Rm=\begin{bmatrix}R_{1,Row[1][1]} & R_{1,Row[1][2]} & \cdots & R_{1,Row[1][c]}\\ R_{2,Row[2][1]} & R_{2,Row[2][2]} & \cdots & R_{2,Row[2][c]}\\ \vdots & \vdots & \vdots & \vdots\\ R_{p\bullet r,Row[p\bullet r][1]} & R_{p\bullet r,Row[p\bullet r][1]} & \cdots & R_{p\bullet r,Row[p\bullet r][c]}\end{bmatrix}$$

$$Qm=\begin{bmatrix}Q_{1,Col[1][1]} & Q_{1,Col[1][2]} & \cdots & Q_{1,Col[1][r]}\\ Q_{2,Col[2][1]} & Q_{2,Col[2][2]} & \cdots & Q_{2,Col[2][r]}\\ \vdots & \vdots & \vdots & \vdots\\ Q_{p\bullet c,Col[p\bullet c][1]} & Q_{p\bullet c,Col[p\bullet c][2]} & \cdots & Q_{p\bullet c,Col[p\bullet c][r]}\end{bmatrix} \quad (6)$$

If we employ the partitioning of H matrix into r rows and c columns of p x p matrices, the R and Q messages in a p x p block can be processed simultaneously. The recent architectures [4, 5, 6 and 7] exploit this property to store messages in the memory partitioned into p independent memory banks and employ p copies of message computation units.

We now represent the R and Q messages in a p x p block as p x 1 vectors

$$\vec{R}_{j,k}=\left[Rm_{1+(j-1)p,k},...,Rm_{l+(j-1)p,k},...,Rm_{p+(j-1)p,k}\right]^T$$

$$\vec{Q}_{k,j}=\left[Qm_{1+(k-1)p,j},...,Qm_{l+(k-1)p,j},...,Qm_{p+(k-1)p,j}\right]^T$$

$$l=1,2,...,p\ \forall j=1,2,...,r, k=1,2,...,c \quad (7)$$

Then R and Q messages in block matrix format are:

$$\vec{R}=\begin{bmatrix}\vec{R}_{1,1} & \vec{R}_{1,2} & ... & \vec{R}_{1,c}\\ \vec{R}_{2,1} & \vec{R}_{2,1} & ... & \vec{R}_{2,c}\\ : & : & : & :\\ \vec{R}_{r,1} & \vec{R}_{r,1} & ... & \vec{R}_{r,c}\end{bmatrix}\vec{Q}=\begin{bmatrix}\vec{Q}_{1,1} & \vec{Q}_{1,2} & ... & \vec{Q}_{1,r}\\ \vec{Q}_{2,1} & \vec{Q}_{2,1} & ... & \vec{Q}_{2,r}\\ : & : & : & :\\ \vec{Q}_{c,1} & \vec{Q}_{c,1} & ... & \vec{Q}_{c,r}\end{bmatrix} \quad (8)$$

Now the Gallager's equations can be written as

$$\vec{R}_{j,k}=\psi\left[\left(\sum_{k=1}^{c}\psi\left(\vec{Q}_{k,j}^{s(k,j)}\right)\right)-\psi\left(\vec{Q}_{k,j}^{s(k,j)}\right)\right].\vec{\delta}_{k,j} \quad (9)$$

$$\vec{Q}_{k,j}=\left(\sum_{j=1}^{r}\vec{R}_{j,k}^{s(j,k)}\right)-\vec{R}_{j,k}^{s(j,k)}+\vec{\wedge}_k \quad (10)$$

$$\vec{\delta}_{k,j}=\left(\operatorname{sgn}\left(\vec{Q}_{k,j}\right).\prod_{k=1}^{r}\operatorname{sgn}\left(\vec{Q}^{s(k,j)}{}_{k,j}\right)\right) \quad (11)$$

$$\vec{\wedge}_k=\left[\wedge\left(1+(k-1)p\right),...,\wedge\left(p+(k-1)p\right)\right] \quad (12)$$

where $\vec{Q}_{k,j}^{s(k,j)}$ ($\vec{R}_{j,k}^{s(j,k)}$) is the modified p x1 vector $\vec{Q}_{k,j}$ ($\vec{R}_{j,k}$), whose elements are circularly shifted in location by the amount $s(k,j)$ ($s(j,k)$).

Say $\vec{A}_j=\sum_{k=1}^{c}\psi\left(\vec{Q}_{k,j}^{s(k,j)}\right), \vec{B}_{k,j}=\psi\left(\vec{Q}_{k,j}^{s(k,j)}\right)$ (13)

$$\vec{C}_k=\sum_{j=1}^{r}\vec{R}_{j,k}^{s(j,k)}, \vec{D}_{j,k}=\vec{R}_{j,k}^{s(j,k)} \quad (14)$$

Now

$$\vec{R}_{j,k}=\psi\left[\vec{A}_j-\vec{B}_{k,j}\right]\vec{\delta}_{k,j} \quad (15)$$

$$\vec{Q}_{k,j} = \vec{C}_k - \vec{D}_{j,k} + \vec{\wedge}_k \qquad (16)$$

Observation

We can observe that the j^{th} block row of R messages is only dependent on the j^{th} block column of Q messages and similarly the k^{th} block row of Q messages is only dependent on the k^{th} block column of R messages. Only one class of messages has to be stored if we schedule the pipeline of the R and Q message computation unit such that the either one of R and Q message units output the block row at once and multiplexing the other units schedule such that it is able to produce the output in block column fashion.

If p Check to Bit serial message computation units, which have internal FIFOs of size $(c\times(r-1)+1) \approx c.r$ are employed, this is approximately equivalent to storage requirement of one class of messages $(p.c.r)$. We do not need any additional memory for storing R and Q messages. By scheduling we can efficiently use the internal memory of the computational units.

3. ARCHITECTURE

For the example (3, 5) - LDPC code of length 1055 described in section 2, $r = 3, c = 5$ and $p = 211$. We can generalize the following discussion to any LDPC code with similar structure.

According to the observation made in Section 2, the pipeline is designed such that Q messages are produced block row wise and R messages are produced in block column fashion. Initially the Q messages are available in row wise as they are set to soft log likelihood information of the bits coming in chunks of p (10). The Q Initializer (Q Init) is an SRAM of size $n+p$ and holds the $\wedge$ values of two different frames. It can supply p intrinsic values to the BCUs each clock cycle and also can simultaneously read p intrinsic values from the channel at the start of iterations of the next frame. The data path of the design is set to 5 bits. ψ and ψ^{-1} are implemented with identical SRAM lookup tables The maximum number of iterations is set to 20 and the iterations will stop when the decoded vector d (using Majority function of Bit to check messages)satisfies the relation $dH^T = 0$.

The p by p interleaver is constructed with two input - two output switches and $\log 2(p)$ stages of $p/2$ switches are used. The Switching Sequence (SS) memory contains the binary sequences to toggle switches in order to produce the shifts in the matrix $S_{3\times5}(2)$. While the interleaver of Q messages will receive sequences column wise (2, 5, 7, 3… 174), the interleaver of R messages will receive sequences row wise (2, 3, 110, 142, 165, 5…174) to complete a decoding iteration (refer to eq.9 and 10).

The Check to Bit processing unit is composed of p serial computation units which computes the partial sum for each block row in a multiplexed fashion to produce the R messages in block column fashion. The registers ps1,ps2 and ps3 correspond to the partial sum for block row 1,2 and 3 respectively.

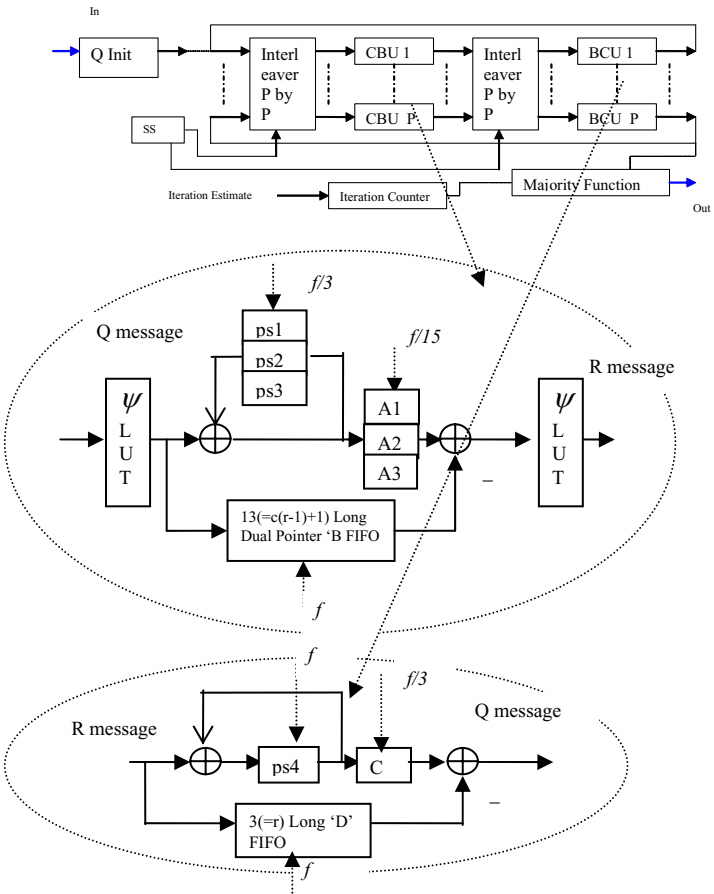

Figure1. Block Diagram of the Decoder Architecture

I	CBU Adders	CBU Sub tractors	BCU Adders	BCU Sub tractors
1	1-15	14-28	17-31	20-34
2	22-36	35-49	38-52	41-55

I=Iteration Number.

Table1. Occupation of Resources for a decoding iteration in terms of clock cycles. (Shown for two iterations.)

I	CBU Adders	CBU Sub tractors	BCU Adders
1	1-15	14-28	17-31
2	19-33	32-46	35-49

Table2. Occupation of Resources for a decoding iteration in terms of clock cycles; with the equivalent implementation of BCU where only required terms are added to reduces idle cycles. This BCU has two adders, three latches, two 2-input mux and one 3-input mux. The

explanation of the architecture is still based on Fig1 for the simplicity.

Clock,I	13,1	15,1	22,1
$ps\vec{1}$	$\sum_{k=1}^{5}\psi\left(\vec{Q}_{k,1}^{s(k,1)}\right)$	$\sum_{k=1}^{5}\psi\left(\vec{Q}_{k,1}^{s(k,1)}\right)$	$\sum_{k=1}^{1}\psi\left(\vec{Q}_{k,1}^{s(k,1)}\right)$
$ps\vec{2}$	$\sum_{k=1}^{4}\psi\left(\vec{Q}_{k,2}^{s(k,2)}\right)$	$\sum_{k=1}^{5}\psi\left(\vec{Q}_{k,2}^{s(k,2)}\right)$	0
$ps\vec{3}$	$\sum_{k=1}^{3}\psi\left(\vec{Q}_{k,3}^{s(k,3)}\right)$	$\sum_{k=1}^{5}\psi\left(\vec{Q}_{k,2}^{s(k,2)}\right)$	0

Table3. Snapshot of partial sum registers in p CBU s operating in parallel to compute p R messages.

The CBU B FIFO corresponds to (13) stores the intermediate computations. Its snapshot at 15th clock cycle is $\left[\vec{B}_{5,3,},\vec{B}_{5,2,}\vec{B}_{5,1,}...,\vec{B}_{1,1,}\right]$.The registers A1, A2 and A3 (which correspond to (13)) latch the ps1, ps2 and ps3 in 14,15 and 16 clock cycles respectively and one of these values (from 14- 28th clock cycle for 1st iteration) will be selected sequentially as one of the inputs to the subtractor and each subtraction operation during this period produces R messages in block column fashion.

The Bit to Check processing unit is composed of p serial computation units which compute the partial sum ps4 for each block row in a sequential fashion to produce the Q messages in block row fashion.

Clock,I	17,1	19,1	31,1
$ps\vec{4}$	$\sum_{j=1}^{1}\vec{R}_{j,1}^{s(j,1)}$	$\sum_{j=1}^{3}\vec{R}_{j,1}^{s(j,1)}$	$\sum_{j=1}^{3}\vec{R}_{j,5}^{s(j,5)}$

Table4. Snapshot of partial sum registers in p BCU s operating in parallel to compute p Q messages

The BCU D FIFO corresponds to (14). Its snapshot at 19th clock cycle is $\left[\vec{R}_{1,1},\vec{R}_{2,1},\vec{R}_{3,1}\right]$ and at 31st clock cycle is $\left[\vec{R}_{1,5},\vec{R}_{2,5},\vec{R}_{3,5}\right]$.The register C (which correspond to (14)) latch the ps4, every three clock cycles and is one of the inputs to the subtractor and each subtraction operation during this period produces Q messages in block row fashion.

4. PERFOMANCE COMPARISON

The rough estimates for total gate count and clock frequency are 2.4e5 and 150MHz respectively when 0.18 micron technology standard cell parameters are used. The architecture is instantiated for one iteration and each iteration takes 18 clock cycles.[Table 2]. The decoder for the 1055 length code (2) can process 440 Mbps for supporting maximum of 20 iterations. The throughput can be improved by instantiating for multiple iterations.

Table5. shows the comparison with the related work. The memory savings are 75% and savings in memory accesses are 66% when compared to [3,4] . When compared to [5,8] the memory accesses are 50% less while the memory requirement is almost the same and this results in better low power characteristic for the proposed architecture. For example [8] reported that the NA-Mm accounts for 50% of their decoder power.

	Yeo [3]	Zhang[4]	Anand [5]	Mansour [8]	Proposed
Mm	$4p.c.r$	$2p.c.r$	$p.c.r$	$p.c.r$	0
Mc	$p.c$	$p.c$	$p.r$	$p.c$	$p.c.r$
NA_Mm	$4p.c.r$	$4p.c.r$	$2p.c.r$	$2p.c.r$	0
NA_Mc	$2p.c.r$	$2p.c.r$	$2p.c.r$	$2p.c.r$	$2p.c.r$

Table5. Memory requirement comparison

Mm: Memory for message storage Mc: Internal Memory in Check to Bit Serial Computational Units NA_Mm: No. of R/W accesses from Mm for a decoding iteration NA_Mc: No. of R/W accesses from Mc for a decoding iteration

5. CONCLUSION

A new decoding schedule for LDPC is presented based on a property of structured LDPC codes. A new architecture based on this decoding schedule is described. The design has reduced memory and correspondingly low power requirements when compared to the existing work. This proposed configuration can be thought of a parallel architecture scaled down by a factor c x r and still having the simple interconnection structure associated with a serial architecture.

REFERENCES

[1] R. G. Gallager, Low-Density Parity-Check Codes, M.I.T Press, 1963. Available at http://justice.mit.edu/people/gallager.html

[2] Blanksby, A.J.; Howland ,C.J, A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder, Solid-State Circuits, IEEE Journal of, Vol.37, Iss.3, Mar 2002 Pages:404-412

[3] Yeo, E.; Pakzad, P.; Nikolic, B.; Anantharam, V., "High throughput low-density parity-check decoder architectures" in proceedings of Global Telecommunications Conference, 2001.Volume: 5, Page(s): 3019 –3024

[4] T. Zhang and K. K. Parhi, "Joint (3, k)-Regular LDPC Code and Decoder/Encoder Design", to appear in IEEE Transactions on Signal Processing. Available at http://www.ecse.rpi.edu/homepages/tzhang/

[5] A. Selvarathinam, G.Choi, K. Narayanan, A.Prabhakar, E. Kim, "A Massively Scalable Decoder Architecture for Low-Density Parity-Check Codes", in proceedings of ISCAS'2003, Bangkok, Thailand.

[6] A. Selvarathinam, G.Choi, K. Narayanan, A.Prabhakar, E. Kim, "A Massively Scalable Decoder Architecture for Low-Density Parity-Check Codes". Journal version in submission.

[7] M. M. Mansour and N. R. Shanbhag, "Low Power VLSI Decoder Architectures for LDPC codes," in proceedings of International Symposium on Low Power Electronics and Design (ISLPED), Monterey, CA, Aug. 2002, pp. 284-289.

[8] M. M. Mansour, M. M. Mansour, and N. R. Shanbhag, "A Novel Design Methodology for High-Performance Programmable Decoder Cores for AA-LDPC Codes," in proceedings of IEEE Workshop on Signal Processing Systems (SiPS), Seoul, Korea, August 2003