International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

SORTING BASED WEIGHTED BIT FLIPPING DECODING TO ACHIEVE HIGH THROUGHPUT IN LDPC CODES

HARSH DURGA TIWARI[1], MEETU TIWARI[2], HONEY DURGA TIWARI[3]

[1]Moderntech, R & D Department, Changwon, South Korea

[2]BK Solutions, R & D Department, Betul, MP, India,

[3]Department of Electronics and Communication Engineering, College, Narsimha Reddy Engineering College, TS, India

E-Mail ID: dr.tiwaridurgaprasad@nrcmec.org

ABSTRACT

Implementation efficient reliability ratio based weighted bit flipping (IRRWBF) decoder for Low density parity check (LDPC) codes provide an excellent trade-off between resource utilization and error correction performance. In this paper, we present a modification in the existing IRRWBF decoding algorithm. It is shown that the time consumption of iterative process can be made independent of code length. The vector implementation using single instruction multiple data (SIMD) technology is presented to achieve high throughput. Based on the specifications provided in IEEE 802.16e wireless standard, the software based implementation of proposed and original decoding scheme achieves 1200% speed improvement.

Keywords: LDPC, Bit flipping, MSD sorting, QC codes

1. INTRODUCTION

Low Density Parity Check (LDPC) codes [1] have shown efficient trade-off between error-correcting performance and decoding complexity by employing bit-flipping based decoding methods [2] - [3]. The IRRWBF [4] algorithm, has especially shown to reduce decoding complexity with error correcting performance superior to other bit flipping algorithms. In IRRWBF decoding, the final aim in each of the iteration is to identify and flip the most unreliable bit associated with largest error term. These bits with highest error value can however be pre-estimated with sufficient accuracy during the first iteration, and the calculation of error bits can be avoided in further iterations. The hard decision bit to be modified can be obtained from the pre-estimated data set and then flipped accordingly. Since, the calculation of error term and finding the bit with highest error value is not done after first iteration, the decoding time and resources are significantly reduced. To demonstrate the effectiveness of the proposed scheme, decoding performance and time was measured on single instruction multiple data (SIMD) architecture based mobile environment. The simulation result shows that the achievable reduction in decoding time was proportional to code length and number of iterations. The summary of

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

symbols used in this paper is given in Table 1.

Table 1. Symbols used in the manuscript

Symbol	Description
H_{mn}	m^{th} row, n^{th} column of parity check matrix **H**, $m = (N - K)$
i	Number of decoding iterations
N	code length
K	information length
r_n	n^{th} bit received from the channel
z_n	Hard decision of r_n
$N(m)$	The set of variable nodes that participate in m^{th} check node
$M(n)$	The set of check nodes in which n^{th} variable node participate
W_c	Column weight
W_r	Row weight
c_r	Code rate $(= N/K) (= 1 - W_c/W_r)$

2. MODIFICATION IN IRRWBF

The IRRWBF algorithm separates the decoding process into four steps: initialization, check node, variable node, and decision steps [4]. For the i^{th} iteration, the number of computations in initialization, check node, variable node, and decision steps are mW_r, mW_r, NW_c, and N respectively. As $c_r = (1 - W_c/W_r)$, it can be shown that for $i >> 1$, the total number of computations is approximately equal to $Ni[(2W_c + 1)]$. This shows that the total number of decoding computations is linearly dependent on N and iteration number i.

2.1 Code length independent decision step

Consider the sorting of all the error terms in the first iteration. Let the sorted list of E_n be given as $e_{n,o}$ where o gives the location of the n^{th} value of E_n value in the sorted list $(e_{n,1} > e_{n,2} > e_{n,3} > \ldots > e_{n,n})$. Hence, in the decision step of the first iteration, the argument n of $e_{n,1}$ is associated with the largest error term. Considering regular LDPC codes, after flipping the bit associated with $e_{n,1}$, W_c number of s_m elements will be changed in the second iteration during the check node step. With each W_c element of s_m, W_r number of E_n elements will be changed in variable node. Hence, in total W_cW_r numbers of E_n elements are changed. By flipping the most unreliable bit in previous iterations the contribution of an unreliable bit for the changed E_n value is reduced and it is safe to assume that $E_n(i^{th}$ iteration) < E_n (i^{th} - 1 iteration) for these changed W_cW_r values. In the previous iteration, as the number corresponding to $e_{n,1}$ has already been flipped, $e_{n,1}$ is no longer the maximum value and the next value in the sorted list must be considered. Hence, if $e_{n,2}$ is not amongst one of the W_cW_r values of E_n which have been changed in second iteration, the z_n bit associated with $e_{n,2}$ is also associated with the largest error ter m for the second iteration. In case $e_{n,2}$ is amongst one of the values that has been modified in the present iteration, then it's current value will be smaller and the next element in the sorted list must be considered for further iterations.

3. PROPOSED BIT FLIPPING ALGORITHM

Based on the discussion in the previous section, the proposed bit flipping algorithm can be summarized as shown in Table II.

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

Table 2. Proposed bit flipping algorithm

Steps	Procedure
Step 1	Perform Initialization, Check node, and Variable node computations same as original IRRWBF decoding for first iteration.
Step 2	Sorting: Sort all the values of E_n in a data set $g_{(n)}$.

Step 3
Iterative Decision:
	i = Current iteration number;
	j = index of the largest error term in the sorted list = 1 for i = 1;
	$U_{(i-1)}$ = set of check nodes updated in previous iterations. $U_{(0)}$ is a null set.
	A_i = set of check nodes expected to be updated in present iteration.
	while $i \leq$ maximum iteration **or** syndrome $\neq 0$ **do**
	if i = 1 **then**
	Flip the bit z_f where f is the argument of $g_{(n)}$.
	$U_{(i)}$ = Set of check nodes that are updated after flipping z_f.
	$j = j + 1$;
	Update W_c check bits (Update syndrome).
	else if $i \geq 2$ **then**
	while $U_{(i-1)}$ and A_j are not disjoint **do**
	$j = j + 1$; Compute A_j;
	end while;
	$U_{(i)} = A_j$;
	Flip the bit z_f where f is the argument of $g_{(n)}$.
	Update W_c check bits (Update syndrome).
	end if;
	end while;

Using a fast sorting algorithm like MSD-radix sort it is possible to sort N number of d digits each, in Nd cycles. Hence, if E_n is represented with d digits, its sorting will take Nd cycles. The number of operations required to construct $A_{l,j}$ can be approximated by number of elements in it, i.e. $W_c W_r$. To find if $A_{l,(\lambda-1)}$ and $A_{l,j}$ are disjoint sets, we need $(W_c W_r)^2$ comparisons in the worst case. Since, $A_{l,(\lambda-1)}$ is already available from previous iterations, its re-computation is not required. Finally, after flipping z_f where f is the argument of $e_{f,j}$, W_c check bits are updated. Hence, the minimum number of operations in each of the iteration ≈ Number of operations required to construct $A_{l,j}$, $(W_c W_r)$ + Number of operations required to check if $A_{l,(\lambda-1)}$ and $A_{l,j}$ are disjoint sets $(W_c W_r)^2$ + Number of operations required to update check bits $(W_c) = (W_c W_r)\ (W_c W_r + 1) + W_c$.

2.1 Reducing number of data comparison

As described previously, the updated E_n values are associated with W_c check nodes. Hence, for any given iteration λ the set of variable nodes updated in previous iteration, $V_{(\lambda-1)}$, are associated with a set of W_c check nodes given by $U_{(\lambda-1)}$. Similarly, error bits expected to be updated, $A_{l,j}$, are associated with a set of W_c check nodes given by $U_{j'}$. If the set of $U_{(\lambda-1)}$ check nodes connected to $V_{(\lambda-1)}$ variable nodes are different from the $U_{j'}$ check nodes connected to $A_{l,j}$ variable nodes, sets $V_{(\lambda-1)}$ and $A_{l,j}$ will also be disjoint. Hence, the sufficient condition to verify if $V_{(\lambda-1)}$ and $A_{l,j}$ are disjoint sets is to check if sets $U_{(\lambda-1)}$ and $U_{j'}$ are disjoint. In this case, the number of operations required to construct $U_{j'}$ may be considered same as the number of elements in it, which is W_c. To find if $U_{(\lambda-1)}$ and $U_{j'}$ are disjoint sets, we need $(W_c)^2$ comparisons in worst case. Since, the $U_{(\lambda-1)}$ is already available from the previous iterations, its re-computation is not required. Hence, the minimum number of operations in each of the iteration ≈ Number of operations required to construct $A_{l,j}$, (W_c) + Number of operations required to check if $A_{l,(\lambda-1)}$ and $A_{l,j}$ are disjoint sets $(W_c)^2$ + Number of operations required to update check bits $(W_c) = (W_c)\ (W_c + 2)$.

4. SIMULATION ENVIRONMENT AND RESULTS

The proposed algorithm was firstly implemented in C and then coded using MMX intrinsic functions to run on Intel® / Marvell® PXA320 (806MHz) CPU. Comparison was made for various code

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

lengths (576, 1440 and 2304, code rate of 1/2) and iterations (5, 10, 30, 50 100) specified in 802.16e standard [5]. For decoding time comparison, 10000 samples of input vectors were taken for each LDPC code. The input samples were pre-stored; the decoding time does not include initialization time and the sample generation time; and hence, the results reflect the true decoding time of the proposed modification.

4.1 Speed and BER performance comparison

To measure the effect of sorting and selective update on the decoding time, the percentage speed increase can be calculated as

$$\text{Percent Speed Up} = \left[\frac{\text{Decoding time (Proposed)}}{\text{Decoding time (IRRWBF)}} - 1\right] \times 100$$

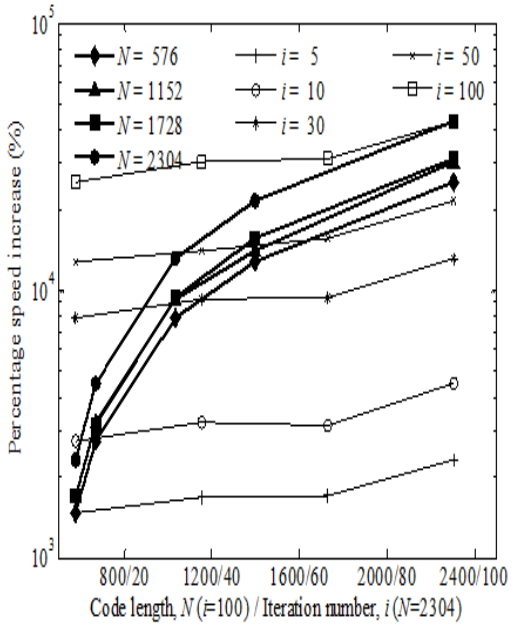

Fig 1. Percentage increase in processing speed for various LDPC codes specified in IEEE 802.16e with different number of decoding iterations as code length, N (or number of iterations, i) is varied

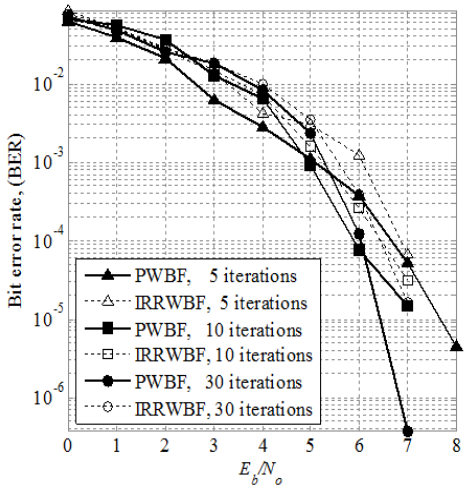

Fig 2. Bit error rate performance comparison of Proposed Weighted bit flipping (PWBF), and Implementation efficient reliability ratio based weighted bit flipping (IRRWBF) [4] decoding for LDPC codes

Figure 1 shows the percentage increase in processing speed with various iteration values, i (or N), when the code length, N (or i), is varied. Figure 2 shows the bit error rate (BER) performance of proposed bit flipping algorithm, and IRRWBF decoding schemes. For applications where high throughput is required, the number of decoding iterations is kept small. The penalty of avoiding En calculations in each of the iteration becomes evident if the number of decoding iterations are made large for medium to high Eb/No ratio. Even with some degradation, the BER performance of the proposed bit flipping codes are always near to the existing bit flipping algorithms.

4.2 Size and decoding time comparison

The comparison of the proposed system with other existing systems is shown in Table 3. Comparison of the decoding time in the proposed and existing SIMD based decoders is presented [6],[7]. Comparison with a hardware-based design is also shown to illustrate the throughput for a given code rate and number of iterations [8]. Although, dedicated-hardware-based designs have

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

higher throughput; they are inflexible and often requires a dedicated and sometimes a bank of parallel aligned memory, used exclusively for LDPC decoding. Software based decoder can operate with variable code length and rate and has no special memory requirement. The multi-core architecture can also achieve reasonably high throughput, however such designs cannot be employed in the mobile environment due to size and power limitations. These comparisons show that with SIMD processing a software-based decoder can achieve a smaller decoding time.

Table 1. Comparison of LDPC decoders

Design	Platform	Code length, N	Code rate, c_r	Decoding iterations, i	Decoding time	Throughput
[13]	NVIDIA 8800 GTX 1.35 GHz OpenGL (GLSL)	1,908	8/9	50	53 ms	
[14]	CELL/B.E., Programming Environment in the 6-Synergistic Processor Elements model (3.2 GHz)	1,024	1/2	50		69.5 Gbps
[11]	(0.18µm) 212 MHz Hardware	2,304	1/2	10		379.23 Mbps
[6]	SPARC Ultra-Enterprise workstation running Solaris 9 OS (SunOS 5.9)	2,000	1/2	50	1396.09 ms	
Proposed	SIMD Processor with 128 bit vector floating point calculation (800 MHz)	2,304	1/2	50	2.475 ms	

6. CONCLUSION

1. In this paper, modification in implementation efficient reliability ratio based weighted bit flipping is presented.
2. It is shown that by flipping the bits in a sorted order the calculation of error terms in each of the iteration can be avoided.
3. The reduction in computation which is proportional to code length and number of iterations is confirmed by the simulation results.
4. The simulation result also shows that the significant reduction in decoding time can be obtained without the loss of error control capability.
5. The implementation highlights the ability to implement real-time LDPC decoders in energy sensitive embedded processor environments which already employ SIMD multimedia acceleration technologies.

REFERENCE

1. R. G. Gallager, Low-Density Parity Check Codes. Cambridge,MA: MIT Press, 1963.
2. Zhang, J., and Fossorier, M.P.C., "A modified weighted bit-flipping decoding of low-density parity-check codes," IEEE Commun. Lett., 2004, 8, (3), pp. 165–167
3. Guo, F., and Hanzo, L., "Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes," Electron. Lett., 2004, 40, (21), pp. 1356–1358
4. Lee, C. H., and W. Wolf, "Implementation-efficient reliability ratio based weighted bit-flipping decoding for LDPC codes," Electron. Lett., 2005, 41, pp. 1356–1358.
5. IEEE P802.16e, D12: Draft IEEE Standard for Local Metropolitan Area Networks. Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, May, 2009.

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

6. Falcao G., Yamagiwa S., Silva V. et al. , "Parallel LDPC decoding on GPUs using a stream-based computing approach," Journal of Computer Science and Technology 24(5): 913-924 Sept., 2009.

7. Falcao G., Sousa L, Silva V, "Massively LDPC Decoding on Multicore Architectures," IEEE Transactions on
9. .

Parallel and Distributed Systems, pp. 1-1, 2010.

8. Lee, J.-Y., and Ryu, H.-J., "A 1-Gb/s Flexible LDPC Decoder Supporting Multiple Code Rates and Block Lengths," IEEE Transactions on Consumer Electronics, Vol. 54, No. 2, MAY, 2008