Technical Note on Iterative LDPC Solutions for Turbo Equalization

Kiran K. Gunnam[1], Gwan S. Choi[1] and Mark B. Yeary[2]
[1]Department of ECE, Texas A&M University, College Station, TX-77843
[2]School of ECE, University of Oklahoma, Norman, OK-73019
Report Date: July 2006

Abstract— Low-Density Parity-Check (LDPC) codes have recently received considerable attention as a next-generation coding technique for communication and storage channels. To meet the demanding error correction requirements at a lower silicon cost, most storage systems manufacturers have started adopting LDPC based error correction systems. This paper proposes an optimized Turbo Equalization System which uses novel statistical buffer management, decoders, and interleaver structures.

***Index Terms*— low-density parity-check (LDPC) codes, decoder architecture, layered decoding, statistical buffering, turbo equalization.**

I. INTRODUCTION

Inter-symbol interference (ISI) channels are encountered in magnetic recording systems, wireless communications and high-speed wire-line modem applications. Maximum a posteriori (MAP), and maximum likelihood (ML) algorithms are generally used for channel detection [1]. For ISI channels with data-dependent noise, pattern-dependent NPML detector (PD-NPML) was proposed by modifying the branch metric computation of ML algorithm and using data-dependent prediction filters. In general, either PD-NPML or pattern dependent noise prediction MAP detectors in conjunction with an iterative LDPC decoder (see [2-6] for more information on LDPC and decoders) can be used in the turbo equalization system [1] to improve the performance.

Our research, see [3-4] and other references to our recent work there in, has introduced the following concepts to LDPC decoder implementation: block serial scheduling, value-reuse, scheduling of layered processing, out-of-order block processing, reconfigurable permuters, dynamic state, speculative computation. In addition, our research introduced the run-time application compiler which has the following concepts: support for different LDPC codes with in a class of codes such as IEEE 802.11n, IEEE 802.16e, array, LDPC etc; off-line re-configurable for several regular and irregular LDPC codes; statistical buffering; run-time change of decoding schedules. All these concepts are termed as On-the-fly computation as the core of these concepts are based on minimizing memory and re-computations by employing just in-time scheduling. The rest of the paper is organized as follows. Section II introduces the system model of turbo equalization that is valid for any communication system that is characterized by ISI channel or ISI channels with data dependent noise. Section III presents the system level design involving new decoder and interleaver structures and statistical buffer management based on queuing theory.

II. SYSTEM MODEL FOR TURBO EQUALIZATION

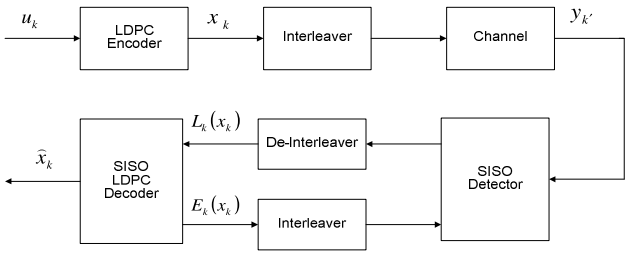

Fig 1. System Model for Turbo Equalization

In the turbo equalization system as shown in Fig. 1, LDPC codeword of length N, obtained by systematic encoding by appending $N-K$ parity bits to the sequence of user information bits of length K. The k-th user information bit is denoted by u_k and the k-th LDPC codeword bit is denoted by x_k. These code symbols are interleaved and the resulting sequence is modulated and transmitted over an ISI channel that introduced additive white Gaussian noise and data-dependent noise. At the output of a front-end timing and matched filtering, the k'-th received sample that corresponds to k-th code word bit is denoted by $y_{k'}$ (called y-sample). Note that k' gives the index of the interleaver output sequence that corresponds to the k-th input to the interleaver.

The performance of the soft-input soft-output (SISO) detection algorithm can be greatly improved if good a priori information is available. The SISO detector computes a posteriori probability log-likelihood ratio $LLR\_APP(x_{k'})$ or $P(x_{k'})$ of each modulation symbol at each time epoch given the received sequence $\{y_{k'}\}_{k'=0}^{N-1}$ and the a priori probabilities from the decoder (also called the extrinsic information from the decoder), $\{E(x_{k'})\}_{k'=0}^{N-1}$. The SISO decoder also computes a

posteriori probability log-likelihood ratios on a code-word basis, $\{P(x_k)\}_{k=0}^{N-1}$ given the a priori probabilities from the detector (also called the extrinsic information from the detector or the channel likelihood ratios when viewed from the LDPC decoder standpoint of view), $\{L(x_k)\}_{k=0}^{N-1}$. The relationship between these messages are given by

$$APP(x_{k'}) = \Pr(x_{k'} \mid y_{k'}) \quad (1)$$

$$P(x_{k'}) = LLR\_APP_k(x_{k'}) = \log\frac{APP(x_{k'}=0)}{APP(x_{k'}=1)} \quad (2)$$

$$P(x_{k'}) = L(x_{k'}) + E(x_{k'}) \quad (3)$$

The interleaver and de-interleaver structures are included into the transmission and receiver systems to spread the burst errors in channel over several modulation symbols and also to disperse the direct feedback effect of iterative updates and can be a global interleaver or local interleaver. Note that the most of the SISO detectors (PDNP-MAP, PD-NPML) that are commonly used operates at the symbol level and SISO LDPC decoder operates at the codeword level. Turbo equalization involves several passes of detection + decoding, each such pass is referred as global iteration. The decoding in a global iteration may involve several LDPC decoding iterations which are referred as local iterations. The reliability messages used in belief propagation (BP)-based offset min-sum algorithm in LDPC decoding 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. 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 based on OMS is described in the following three steps as given in [2-4]:

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

$$R_{mn}^{(i)} = \delta_{mn}^{(i)} \max\left(\kappa_{mn}^{(i)} - \beta, 0\right), \quad (4)$$

$$\kappa_{mn}^{(i)} = \left|R_{mn}^{(i)}\right| = \min_{n' \in \mathrm{N}(m)\setminus n} \left|Q_{n'm}^{(i-1)}\right|, \quad (5)$$

where β is a positive constant and depends on the code parameters . 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 $m \in \mathrm{N}(n)$,

$$Q_{nm}^{(i)} = L_n^{(0)} + \sum_{m' \in \mathrm{M}(m)\setminus m} R_{m'n}^{(i)}, \quad (6)$$

where the channel log-likelihood ratio of bit n is $L_n^{(0)} = L(x_n)$ and this is communicated from the detector for each global iteration.

Step 3. *Decision: for final decoding*

$$P_n = P(x_n) = L_n^{(0)} + \sum_{m \in M(n)} R_{mn}^{(i)}. \quad (7)$$

The extrinsic message is computed as follows,

$$E_n = E(x_n) = \sum_{m \in M(n)} R_{mn}^{(i)}. \quad (8)$$

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 $\hat{x}H^T = 0$ where H is the LDPC parity check matrix, the decoding process is finished with $\hat{x}_n$ as the decoder output; otherwise, repeat steps (1-3) till maximum number of iterations. The above equations describe non-layered decoding. For the layered decoding [3-6], (7-8) is updated after doing the check node updates in a block row or layer of LDPC matrix and (6) for the next layer is computed based on latest check node messages of the previous layer. At the start of detection during the first global iteration, the extrinsic message E from LDPC decoder is set to 0. The E message from LDPC decoder is communicated to the detector at the end of all the local iterations i.e. once per each global iteration. The Final State (FS) in the LDPC decoder contains the compressed information of check node messages in the form of least minimum, second least minimum, index for least minimum, cumulative sign of R messages and the sign messages of R messages for each check node [3-4].

III. System Level Architecture for Turbo Equalization

Fig 2. gives the proposed system level architecture for turbo equalization. The innovation here is the novel and efficient arrangement of queue structures such that we would get the performance of a hardware system that is configured to run h(which is set to 20 in the example configuration) maximum global iterations while the system complexity is proportional to the hardware system that can 2 maximum global iterations. The y-sample data for each arriving packet is buffered in Y queue. Since the data comes at deterministic time intervals in a real-time application, the arrival process is D and the inter-arrival time is T. The overall processing/service time for each packet is variable and is a general distribution G. Assume that 4 y-samples per clock in each packet are arriving and the

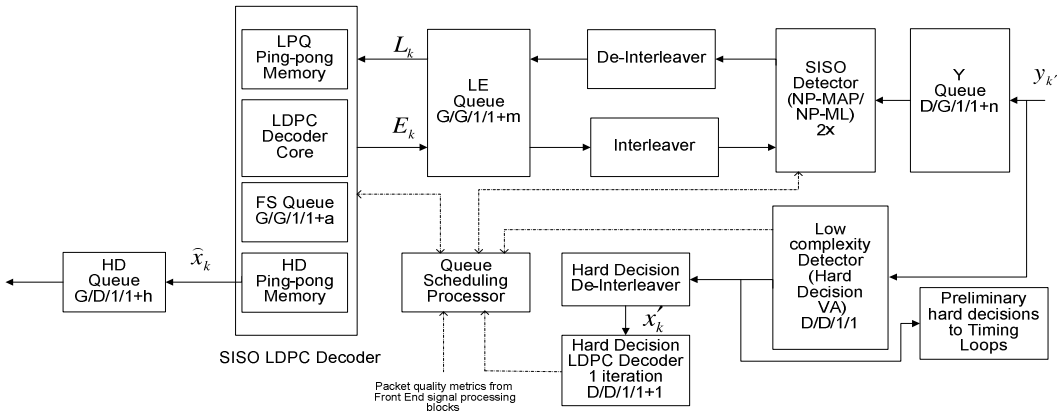

Fig 2. Proposed System Level Architecture for Turbo Equalization

packets are coming continuously. In real-time applications, we need to be able to process 4-samples per clock though some latency is permitted. The primary data-path contains one SISO LDPC decoder and one SISO detectors. The LDPC decoder is designed such that it can handle the total amount of average iterations in two global iterations for each packet. The SISO detector is in fact two detector modules that operate on the same packet but different halves of the packet thus ensuring one packet can be processed in 50% of the inter-arrival time T. Each detector processes 4 samples per clock cycle. Thus both the detector and the LDPC decoder can sustain maximum of two global iterations per each packet if no statistical buffering is employed. In the last successful global iteration the LDPC decoder does the variable number of local iterations. The left-over LDPC decoder processing time is shared to increase the number of local iterations in following global iterations for the next packet. For each packet, at least one global iteration is performed and the distribution of required global iterations follows a general distribution that heavily depends on the signal to noise ratio. Note that if the system needs more than 2 guaranteed global iterations, then we need to increase the detection capability from 2x4 samples/clk to mx4 samples/clk.. For instance, if we need m guaranteed global iterations, then we will make the detector as *m*x detector, so that the SISO detector is in fact 6 detector modules that operate on the same packet but m different segments of the packet thus ensuring one packet can be processed in 1/m of the inter-arrival time T. Also we may need to adjust the decoder edge parallelization to meet the desired number local iterations for each global iteration.

The de-interleaver is such that it is similar to a local de-interleaver used in turbo-code based systems. However there are some optimizations we can perform to improve the performance of the local interleaver. The local de-interleaver after the SISO detector can buffer up to *p* L samples (where *p* is the circulant/block size in H matrix), *p/m* samples will be coming from each detector of *m*x detector. The local de-interleaver can do random interleaving or any other pseudo or algebraic interleaving of *p* x *p*. Now these *p* L samples can be mapped to any unmapped circulant or block of size *p* in H matrix in the LDPC domain and written to LE memory and then transferred to LPQ memory to start the decoding. So while we are doing local de-interleaving similar to turbo code based system, we are doing another level of global mapping such that the *N/p* de-interleaved local chunks can belong to *N/p* different circulants (blocks) of the H matrix-thus achieving a global spread of the de-interleaved chunks to different portions in the codeword. Note that we can have up to *N/p* different local mappings for each of the *N/p* chunks. The advantage of proposed interleaver(union of *N/p p* x *p* local de-interleavers followed by *N/p* x *N/p* global interleaving at the chunk level which is simply address look up for the memories while writing into the LE memory) is it can achiever similar interleaver properties(spread and burst error correction) as of *NxN* global interleaver. If the circulant/block size *p* is 1(i.e. the H matrix is a random H matrix), then there is no local interleaving and only global mapping of size *N* x *N*. On the contrary, we can also consider this as H matrix having block size *p* is same as N, then there is local interleaving of size *p* x *p* and there is no global mapping. The de-interleaving operation is inverse of interleaving operation. After the decoding, E samples will be available from the decoder into LE memory. The interleaver will readout *p* E samples corresponding to a circulant from LE memory such that these samples could be fed in-order such that the SISO detector can consume of them on-the-fly. Since we may have *m* detectors operating in parallel, the chunk of size *p,* would be split into *p/m* sub-chunks and fed to different detectors. In the Figure 2, we will have m=2, so we have two different detectors operating on two halves of the same packet. Say we have the circulant size *p* is equal to 128, *N*=40960, *N/p*=320. Say we just started detection at the rate of 8 samples/clk from SISO detector (4 samples/clk from first detector operating on the first half of packet and 4 samples/clk from second detector operating on the second half of packet). Now we will accumulate 64 L samples from first detector and 64 L samples from second detector and form a chunk of size 128 L samples. Now we will do 128x128 random interleaving on this first chunk. Then we will store this chunk (i.e. interleaved 128 L samples) numbered 1 into as LDPC circulant number 30 into LE memory as an example. In general interleaved chunk numbered *kk* can be written as circulant numbered *dd*. The mapping is given by a global mapping table or address look up table of size 320. Row *kk* in this table gives the circulant number, *dd* to which address the interleaved chunk *kk* is mapped to. The optimization of *N/p p* x *p* local interleavings and *N/p* x *N/p* global mapping is an NP-complete problem and can be solved as a computer search optimization for given SNR and burst error capabilities.

The secondary data path contains the low complexity detector based on hard decision Viterbi algorithm, a hard

decision interleaver followed by hard decision LDPC decoder that is sized for doing only one iteration. The secondary path thus does one reduced complexity global iteration and operates on the incoming packets immediately. The secondary path's immediate preprocessing of incoming packets has two uses: 1) it can generate preliminary decisions immediately (with a latency equal to T) to drive the front end timing loops thus making the front end processing immune from the variable time processing in the primary data path 2) it can generate quality metrics to the queue scheduling processor. The low complexity detector is in the arrangement D/D/1/1 according to Kendall Notation [7]: the arrival times are deterministic, the processing time/service times are deterministic, one processor and one memory associated with the processor. For a good introduction on queuing theory and notations, please refer to [7]. The low complexity decoder is in the arrangement D/D/1/1+1 – this is similar to the low complexity detector except that there is one additional input buffer to enable the simultaneous filling of the input buffer while the hard decision iteration is being performed on a previous packet. Note that LDPC decoder needs the complete codeword before it can start the processing. The queue scheduling processor takes the various quality metrics from the secondary reduced complexity data path as well as the intermediate processing from the primary data path. One example of a quality metric is the number of unsatisfied checks from LDPC decoder. All the queues in the primary data path are preemptive such that the packets are processed according to the quality metric obtained through preprocessing.

The LPQ memory in LDPC SISO decoder which is configured as layered decoder is used to buffer the L messages from the detector before the start of local iterations. The same buffer space is overwritten with the Q messages that are internal to the LDPC decoder during the decoding process. At the end of all the local iterations of a global iteration, the same buffer space is overwritten with the P messages that are internal to the LDPC decoding process. The LPQ memory is double buffered such that while the decoder uses one of the memories as Q memory for one packet, another packet's L messages are being filled in another memory from the detector. Note that the bandwidth of this memory is dictated by the parallelization requirements of LDPC decoder and these memories are very wide and thus occupy more area and consume more power due to the large number of internal memory accesses in LDPC decoder. This is the reason why we can not afford statistical buffering on LPQ memory.

The LE and Y queues can hold several packets and this memory bandwidth is proportional to the number of samples processed by the SISO detector thus making it a good choice for statistical buffering. Assuming 4-samples per clock input rate, for the Y queue, the read bandwidth is 8 samples/clock and write bandwidth is 4 samples per clock. For the LE queue, the read bandwidth is 8 samples/per clock and the write bandwidth is 8 samples per clock from the de-interleaver on the detector side; the read bandwidth is 8 samples/per clock and the write bandwidth is 8 samples per clock from decoder. The LE memory holds the L values from the detector after the detection is over and this is preserved till the LDPC decoding is finished and the LPQ memory is updated with the P values. Now the E values are computed by subtracting L values from P values These E values overwrite the space occupied by L values if further global iteration(s) is necessary. The E values are supplied to the detector and preserved to compute L values while the detector computes its own update of P messages. The L computation is given by P-E. LE queues will hold either L or E message for a packet. If decoding is successful, the space for the packet in both LE and Y queue is released. Due to the condition that we always retain the slot till the decoding is sucessful, the only processor thar can can retire a packet from LE and Y queue is LDPC decoder. Thus the LE queue is modeled as G/G/1/1+m whose arrival time and serices times are general distibutuions, 1 indicates that there is one processor that can complete the service. Here m indicates the maximum number of packets paused and waiting to be resumed . The Y queue is modeled as G/G/1/1+n similar to LE queue and also goverened by general distributions(note that using G means general distribution and each queue has a unique general distriibution). Here n indicates the maximum number packets either in pause state waiting to be resumed or waiting to be processed for the first time.

The memory bandwidth of hard decision memory (HD memory) is proportional to the LDPC decoder's parallelization and is large. So the HD memory is not a suitable candidate for statistical buffering. Instead we allocate double buffering for HD memory and do the statistical buffering on a lean (low bandwidth) HD queue. Note the write bandwidth is 4 bits/clock and the read bandwidth is also 4 bits/sec . While HD queue is filled by one of HD memories and another HD memory is being used internally for the decoder processing. The statistical buffering of HD memory is to enable keeping the packets that are already decoded as the packets may complete process out-of-order The HD queue is thus characterized as G/D/1/1+h as the arrival process is a general distribution while the serving time (i.e. the supply of the decoded packets to the sink) is constant and deterministic in order to meet real-time requirements. Here h determines the maximum latency for processing a packet that is measured as the difference between arrival time of y-samples of a packet to the serving time of hard decisions to the sink by the HD queue. The FS queue is modeled as G/G/1/1+a as both the service time and arrival time are general distribution. 1 indicates one LDPC processor and 1+a indicates the number of additional packet that can have their Final state preserved. Thus the variable a indicates how many number of packets can be paused while being processed and can retain the final state from the previous processing. All the queues are preemptive such that the packets are processed according to the quality metric obtained through preprocessing.

IV. REFERENCES

[1] R. Koetter, A.C. Singer, and M. Tuechler, "Turbo Equalization," IEEE Signal Processing Magazine, invited paper, Special Issue on Graphical Models,Volume: 21 Issue: 1 , Pages:67 – 80, Jan. 2004.

[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] K. Gunnam, W. Wang, E. Kim, G. Choi, and M. Yeary, "Decoding of quasicyclic LDPC codes using an on-the-fly computation," Accepted for Asilomar Conference on Signals, Systems, and Computers, 2006.

[4] K. Gunnam, G. Choi, M. Yeary, and M. Atiquzzaman, "VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax," Texas A&M Technical Report.

[5] 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.
[6] H. Sankar and K.R. Narayanan, "Memory-efficient sum-product decoding of LDPC codes," IEEE Transactions on Communications, vol.52, no.8pp. 1225- 1230, Aug. 2004.
[7] D. Gross ,J. Shortle,J.Thompson, C. Harris, *Fundamentals of Queueing Theory*, 3rd Edition, John Wiley, 1998.