BCSIR

Available online at www.banglajol.info

Bangladesh J. Sci. Ind. Res. **47(3)**, 327-332, 2012

BANGLADESH JOURNAL OF SCIENTIFIC AND INDUSTRIAL RESEARCH

E-mail: bjsir07@gmail.com

Hardware implementation issues of turbo decoders

M. S. Islam[a], M. A. Quaium[b]*, M. Morshed[a] and R. C. Roy[c]

[a]Department of Electrical and Electronic Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh, [b]Department of Electrical Engineering, Technical University of Delft, The Netherlands and [c]Pilot Plant and Process Development Center (PP & PDC), BCSIR, Dhaka, Bangladesh.

Abstract

This paper gives a general overview of the implementation aspects of turbo decoders. Although the parallel architecture of the turbo code is emphasized, the serial concatenated convolutional codes for the turbo decoder are discussed too. Considering the general structure of iterative decoders, the main features of the soft input and soft output algorithm, which are the heart of a turbo decoder, are observed. The efficient parallel architectures of turbo decoders are shown which allow high speed implementation. Apart from these, implementation aspects like quantization issues and stopping rules to increase the throughput as well as an evaluation of the various turbo decoders are discussed. Finally, we suggest a number of solutions to overcome the implementation issues as well as the complexities without affecting the high throughput rate.

Key words: SCCC, SOVA, PCCC, SISO, LLR etc.

Introduction

The turbo codes first drew the attention of the channel coding research community in 1993 at a seminar. The performance claimed in that seminal paper was soon confirmed with a practical hardware implementation (Communication, Nov. 1993). And the realm of turbo codes began from that very day.

The performance of turbo codes and their suitability for practical implementation led them to be adopted in various communication standards in the late 90s. They were chosen in the telemetry coding standard by the Consultative Committee for Space Data Systems (CCSDS) (CCSDS, Sep. 2003) and for the medium to high data rate transmissions in the third generation mobile communication 3GPP/UMTS standard (Jungu *et al.,* 2000). To enable the broadband interactive satellite and terrestrial services, they have been adopted as a part of the Digital Video Broadcast Return Channel Satellite and Terrestrial (DVB-RCS and DVB-RCT) links (DVB, 2000 and DVB, 2001). That was not the end, more recently; they were also selected for the next generation of 3GPP2/cdma2000 wireless communication systems (3GPP2, Feb. 2004) as well as for the IEEE 802.16 standard (WiMAX) (IEEE, Nov. 2004) which was intended for broadband connections over long distances. Besides the parallel codes a serial turbo code was also adopted in 2003 by the European Space Agency for the implementation of a very high speed (1 Gb/s) adaptive coded modulation modem for satellite applications (Boutillon *et al.,* 2007).

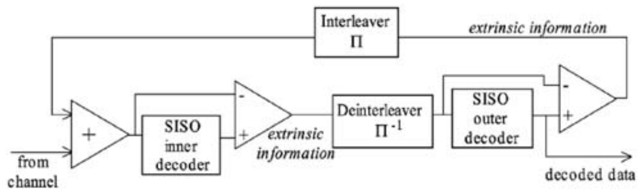

Fig. 1: General principle of the turbo decoding in logarithmic domain

This paper discusses the various implementation methods and aspects of Turbo Decoders. The high speed implementation considering the parallel architectures of the turbo decoders are presented. The implementation issues like stopping rules and quantizations are also discussed. This paper is organized as following. In section II, the turbo codes in general were introduced. Section III discusses the solution to increase the throughput of the decoder in the parallel architecture. Section IV discusses the various stopping rules. In section V, the quantization issues are discussed. Section VI discusses the complexities faced in iterative decoding. In Section VII, some additional concepts and suggestions are discussed.

*Corresponding author. E-mail: adnan.quaium@ubuntu.com

Turbo Codes

Turbo codes are variants of convolutional codes. As stated by Forney (Boutillon *et al.,* 2007), concatenation is a method of building long codes out of shorter ones in order to resolve the problem of decoding complexity by breaking the required computation into manageable segments according to the divide and conquer strategy. Turbo codes, also known as parallel concatenated convolutional codes (PCCC), are based on a parallel concatenation of two recursive systematic convolutional codes separated by an interleaver. They are called turbo in reference to the analogy of their decoding principle with the turbo principle of a turbo-compressed engine, which reuses the exhaust gas in order to improve efficiency. The turbo decoding principle calls for an iterative algorithm involving two component decoders exchanging information in order to improve the error correction performance with the decoding iterations. This iterative decoding principle was soon applied to other concatenations of codes separated by interleavers, such as serial concatenated convolutional codes (SCCC), sometimes called serial turbo codes, or concatenation of block codes, also named block turbo codes (Boutillon *et al.,* 2007). Fig. 1 shows the general principal of the turbo decoding method.

Parallel Architecture of the Turbo Codes

There are three solutions to increase the throughput of the decoder: increasing the parallelism of the decoder, increasing the clock frequency, and decreasing the number of iterations. The increase of parallelism can be obtained at all the levels of the hierarchy of the turbo decoding algorithm first at the turbo decoder level, second at the SISO level, third at the half iteration level and finally, at the trellis stage level.

A. Codeword Pipeline and Parallelization

In this method, the $2n_{it}$ numbers processors work in a linear systolic way, while the first one processes the first half iteration of the newest received codeword of index k, the second one processes the second half iteration of the previous received codeword (index k - 1), and so on, up to the $2n_{it}$ th processor that performs the last iteration of the codeword of index k - $2n_{it}$. Once the processing of half iteration is finished, all codewords are shifted in the linear processor array (Boutillon *et al.,* 2007).

B. Parallel SISO Architecture

In this method, several independent SISO decoders work on the same codeword. To do so, the frame of size N is sliced into P slices of size $M=N/P$ and each slice is processed in parallel by P independent SISO decoders. This technique implies two types of problems: the problem of data dependency, which is solved by relaxing the constraint of performing the entire forward (respectively, backward) processing within a single iteration (Blankenship *et al.,* Jun. 2005 and Giulietti *et al.,* Feb. 2002); and, the problem of the memory collisions, which can be solved either in execution stage or in compilation stage or in design stage. One of the main issues of parallel architecture is dealing with the efficient usage of all available computational resources.

C. Parallel Trellis Stage

As the both the forward and backward recursion contains a loop, it is not possible to increase the parallelism directly. However, there is a technique, known as trellis compacting, in which the conventional trellis is reconstructed by grouping two consecutive trellis stages in a single one. This trellis compaction leads to an equivalent trellis of the trellis of the double binary code.

D. Increase of Clock Frequency

To increase the clock frequency, the critical path of the turbo decoder should be reduced. The critical path is in the forward or backward recursion loop. There are a few solutions to reduce this path directly, such as, using of a fast adder, reducing the number of bits to code and, delaying of one cycle if the log-map algorithm is implemented.

Stopping Rules and Buffering

By introducing some stopping criterion, the throughput of the decoder can be increased to exploit the randomness of the number of required iterations (Matache *et al.,* Aug. 2000). The most efficient and simple stopping rules are:

A. Hard Rule 1

The signs of the LLRs at the input and at the output of a constituent SISO module are compared and the iterative decoder is stopped if all signs agree.

B. Hard Rule 2

To improve the reliability of the stop rule, the previous check has to be passed for two successive iterations

C. Soft Rule 1

The minimum absolute value of all the extrinsic LLR at the output of a SISO is compared against a threshold.

D. Soft Rule 2

The minimum absolute value of the entire total LLR is compared against a threshold.

Quantization Issues in Turbo Decoders

The hardware complexity increases linearly with the internal bit width representation of the data. For that the minimum bit width internal representation that leads to an acceptable degradation of performance need to be formulated. Without significant degradation of the performance compared to a classical DSP application, the internal precision of a turbo-decoder can be very low (Boutillon *et al.*, 2007).

A. Internal Precision of a Turbo Decoder

If x is the received signal and σ is the variance of white Gaussian noise then LLR λ (c_l:I) can be written as follows.

$$\lambda(c_l : I) = \frac{2\bar{x}}{\sigma^2}$$

The Quantization value of $\lambda(c_l{:}I)$ can be given by a Quantification Function Q, defines as

$$x_Q = Q(x) = sat\left(\left[x.\frac{2^{b_{LLR}-1}}{A}+0.5\right], 2^{b_{LLR}-1}-1\right)$$

where *sat* (*a, b*) =*a*, if a belongs to [-*b, b*] , and sat(*a,b*) = *sign(a)xb* , otherwise; data are quantized between [-*A,A*]. If *A* is very large, most of the input will be quantized by a zero value, i.e., an erasure, and the decoding process fails. And if *A* is too small then saturation happens most of the time and the soft quantization would thus be equivalent to a hard decision. So for a given code rate and a given SNR there is an optimal value of *A*.

Input values of $\lambda(c_l{:}I)$ depends on channel observation *A*, as well as the noise variance σ^2 (variance of the SNR of the signal). When the max-log-MAP algorithm is used, the estimation of SNR is not necessary. When the log-MAP algorithm is used, the real standard deviation σ is replaced by the maximum value σ so that it produces a Bit Error Rate (BER) or a Frame Error Rate (FER) acceptable for the application. The number of bits b_{ext} to code the extrinsic message can be deduced from b_{LLR}. Once b_{LLR} and b_{ext} are chosen, the number of bits b_{fm} to code the forward recursion metrics α and the backward recursion nodes β can be derived automatically. In (Gross *et al.,* Aug 1998) it is shown that, for forward recursion, if the v is the memory depth and Γ.is the function of code, band b then at any time l, It is also shown

$$\max_S(\alpha_l(s)) - \min_S(\alpha_l(s)) < v.\Gamma$$

that if the $min_s(\alpha(s))$ is maintained zero then only b_{fm} is sufficient to code the forward metrics. But the problem is to maintain a zero level, additional hardware is required which increases the critical paths for forward recursion and complexities. A more efficient solution is to replace those complex systematic operations by the subtraction of a fixed value when needed. The subtraction is simply realized by setting all the MSB to zero.

B. Practical Implementation of MAX Algorithm*

A block diagram of the MAX* operator is shown according to the definition in (Matache *et al.,* Aug. 2000). A lookup table performs the computation of the correcting factor given by the following equation.

$$f(x) = \ln\left(1 + e^{-|x|}\right)$$

The maximum value of this function is *ln (2)* and the function decreases rapidly toward zero. A working diagram of MAX* operator is shown in the Fig. 2.

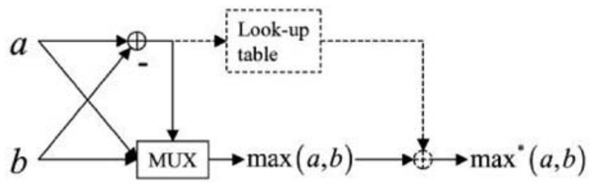

Fig. 2: Block diagram of a MAX* operator

C. Rescaling of Extrinsic Message

The use of the max-log-MAP algorithm leads to an overestimation of the extrinsic message, which in turns decreases the performances of the turbo decoder. This can be significantly avoided if the overestimation of the extrinsic message is compensated, on average, by a systematic scaling down of the extrinsic message between two consecutive half iterations. This scaling factor is also favorable with respect to the de-correlation of the extrinsic messages.

D. Method of Optimization

Various parameters like number of bits of quantization, maximum number of iterations, scaling factors of the extrinsic message, BER and FER, impact both the performance and the complexity of the turbo decoder. All those parameters interact in a nonlinear way. So finding a good performance-complexity trade off is a very complex task. In order to have an accurate estimation of the BER a CPU extensive Monte

Carlo simulation is required. To avoid the simulations, the proposed methods are:

1. Define the space of the search by defining the range of search for each parameter of the decoder as well as define a model of complexity for each parameter along with the maximum allowable complexity of the design.
2. Define the proper worst case configuration.
3. Using this configuration, perform a Monte Carlo simulation at the SNR of interest. Each time a received codeword fails to be decoded, store the codeword in a set S then stop the process when the cardinality of the set S is high enough.
4. Perform an optimization in order to find the set of parameters that minimize the BER (or the (FER) over the set S.
5. Perform a normal Monte Carlo simulation in order to verify a posteriori the real performance of the selected parameters.
6. Go to Step 1 with a different scenario of optimization if needed.

Evaluation of Complexities of Iterative Decoders

For a high level evaluation of the complexity of the decoder, the architecture is shown in Fig 3. Where the shaded blocks are processors and white blocks are memories. The summary of the whole process is:

1. Initialize the inner memory to null messages.
2. Apply the first set of constraints A using the EXT and LLR, write the updated messages in EXT.

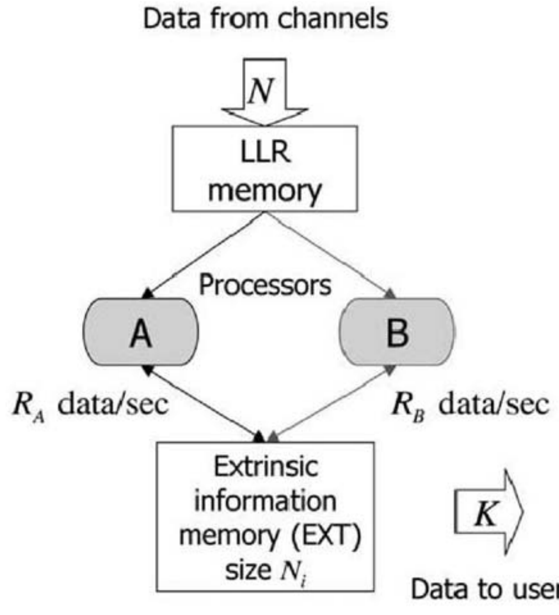

Fig. 3: General architecture of iterative decoder

3. Initialize the inner memory to null messages.
4. Apply the first set of constraints *A* using the EXT and LLR, write the updated messages in EXT.
5. Apply the second set of constraints *B* using the EXT and LLR, write the updated messages in EXT.
6. Iterate until some stopping criterion is satisfied or the maximum number of iterations is reached.

To gain control on the trade off between area and throughput of the decoder the number of elementary operations required for decoding one information bit per iteration as a function of the main design parameters of the code is considered to be focus, which is indicated as *C*. If C_{dep} is the number of deployed operators running at frequency *f*, and N_{it} is the number of required iteration, then the throughput *T* of the implemented decoder can be well approximated by

$$T = f\left(\frac{C_{dep}}{N_{it}C}\right)$$

A. LDPC

In case of LDPC, each variable node processor with degree d_v requires $2d_v$ sums to compute the updated message. So, summing up over all possible nodes, the variable node processing requires two sums per edge. By summing up, the following complexities are found:

$$c = \begin{Bmatrix} \dfrac{2\eta}{R} \\ 6\left(\dfrac{\eta-2}{R}+2\right) \end{Bmatrix}$$

Here *R* is the rate of the code, and *n* measures the density of the LDPC parity check matrix.

B. PCCC and SCCC

The complexity of PCCC (parallel concatenated convolutional codes) and SCCC (serial concatenated convolutional codes) is strictly related to the complexity of their constituent SISO decoders. Here, *k/n* rate is considered for binary constituent encoders (Boutillon *et al.,* 2007). Two versions of SISO has been considered here:

1. The inner SISO, which is used in PCCC and as the inner SISO for SCCC, gets messages on information and coded bits and provides messages on input bits.

2. The outer SISO, which is used as outer SISO in SCCC, gets messages only on coded bits and provides updated messages on both information and coded bits.

After determining the complexity per information bit for the inner and outer SISO decoders C_I and C_O, and if the r_o is rate of the outer encoder for SCCC, the complexity of the PCCC and SCCC can be evaluated as

$$C_{PCCC} = 2C_I$$
$$C_{SCCC} = \left(C_O + \frac{1}{r_O} C_I\right)$$

C. Memory Requirements

The memory requirements of an iterative decoder is the sum of memory required for the storage of channel messages, which is *N* for all types of decoders, and the memory for the storage of the extrinsic messages, which depends on the encoding scheme.

D. Nonbinary Decoders

The main consequence of using Nonbinary Decoders is that messages are no longer scalars but vectors of dimension equal to the cardinality of the used alphabet minus one. The dimension of message memory must then be increased accordingly. So the performance is also different.

Suggestions for Implementation Issues

The hardware implementation of turbo decoding still faces some issues. However, theoretically some solutions can be suggested to overcome these issues. In this section we suggest some solutions to mitigate the implementation issues. Although these solutions are very much theoretical, they can be implemented practically.

A. Using SOVA

To avoid the computational complexities (to reduce the hardware complexities as well), Soft Output Viterbi Algorithm (SOVA) can be applied instead of MAP algorithm. MAP algorithm tries to minimize the code word error by maximizing the probability, while SOVA attempts to maximize the a-posteriori probabilities (APP) of the individual bit (Hagenauer *et al.,* 1989). MAP algorithm takes all paths into consideration and generates the sum of probabilities of all paths in the estimation. On other hand SOVA produces the soft output by considering only two maximum likelihood (ML) paths, which eventually shows less complexity. Though SOVA may not perform as efficient as MAP over the AWGN or Fading Channel, one can consider the improved algorithm of SOVA (Chuan Xiu Huang *et al.,* 2004) in that perspective.

B. Additional Stopping Rules

A part from the Hard Stopping Rules and Soft Stopping Rules, a rule based on detecting erroneous decoded sequences using an outer cyclic redundancy check (CRC) code can be applied to hard decoded bits (Matache *et al.,* Aug. 2000). In this rule, a separate error detection code, (such as a CRC code) can be concatenated as an outer code with an inner turbo code in order to flag erroneous decoded sequences. The condition for stopping with this rule is satisfied whenever the syndrome of the CRC code is zero. Also a finite termination condition for all rules can be applied. So that if the maximum number of iterations is reached before the stopping rule is satisfied, one may use this condition to flag detected errors (Rovini *et al.,* 2006).

C. Simplifying the Offset Function

As the Log-MAP algorithm is a logarithmic domain description of the MAP algorithm, neglecting the offset term, we get the Log_Max algorithm. In interference limited systems like WCDMA, this approximation may result in a capacity loss of as much as 10%, and hence we need to include also the offset term. The drawback of including the offset term is that it makes the algorithm sensitive towards SNR mismatch. Since the offset term is a nonlinear function, it is typically approximated by a set of lookup tables. However, for ASIC implementation this implies a need of high speed memory. Instead of this a linear approximation (the Log-Lin algorithm) of the offset term can be used (Jungu *et al.,* 2000). It has been seen that the Log-Lin algorithm achieves almost the same performance as the Log-MAP algorithm.

*D. Simplified MAX**

Another way can be the most commonly used MAP algorithm in turbo decoding is the BCJR algorithm. The MAP decoders make optimum symbol-by-symbol decisions, as well as providing 'soft' reliability information which is necessary in concatenated decoding systems such as turbo decoders. BCJR algorithm suffers several shortcomings which make it unsuitable for VLSI implementation, namely the requirement of multiplications and exponentiations. So a simplification of the MAX* operation (Gross *et al.,* 1998) can be introduced for the Log-BCJR algorithm which replaces the lookup table with a constant value. The simulations (Gross *et al.,* 1998) show that turbo code performance is not adversely affected by the modification.

E. Rescaling technique

A more elaborated rescaling technique has been shown in (Hekstra 1989). The rescaling operation can simply be avoided by using the modulus arithmetic. In the rescaling scheme, at each iteration the minimum metric is subtracted from all metrics. The use of two's complement arithmetic is proposed as an alternative to the rescaling method, which surprisingly avoids any kind of rescaling subtractions. Avoiding the rescaling has its advantages in implementation such as hardware savings, a speedup inside the metric update loop (which is critical to the decoder's computational throughput). As a result, the use of two's complement arithmetic to accommodate metric overflow in the Vitterbi algorithm offers significant advantages in implementation, in terms of design simplification and computational throughput.

Conclusion

In this paper, we have presented an overview of implementation issues for the design of turbo decoders in the context of the concatenation of convolutional codes. We discussed that how the speed of decoding can be increased by raising the decoder clock frequency, increasing the use of parallel hardware, and judiciously limiting the number of decoding iterations. Hardware complexities were discussed and assessed too. In addition some suggestions to overcome these issues were also provided. We have focused on the different methods allowing the throughput of a turbo decoder to be increased. We have particularly investigated parallel architectures and stopping criteria. Nowadays, either in software or hardware, low throughput (below 10 Mb/s) turbo decoders have been widely implemented. And those are commercially available too. But some challenges in future are yet to come, on which further research are widely welcomed. Among the main challenges in the years to come, low energy consumption receiver design will represent a crucial one. To gain a significant progress in this field requires a real technological breakthrough.

References

Boutillon E, Douillard Cand Montorsi G (2007). Iterative decoding of concatenated convolutional codes: Implementation issues, Proceedings of the IEEE, **95**(6): 1201 - 1227.

Blankenship TK, Classon B and Desai V (May. 2002). High-throughput turbo decoding techniques for 4G, in Proc. Int. Conf. 3G Wireless and Beyond, San Francisco, CA, 137-142.

Chuan Xiu Huang, Ghrayeb A (Sep. 2004). An improved SOVA algorithm for turbo codes over AWGN and fading channel, Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium, 2, 1121 - 1125.

Communication (Nov. 1993). Comatlas, CAS5093: Turbo Encoder/Decoder, data-sheet. **37**(11).

CCSDS (Sep. 2003). Recommendation for Space Data System Standards. TM Synchronization and Channel Coding, 131.0-B-1, Blue Book.

DVB, Interaction Channel for Satellite Distribution Systems, ETSI EN 301 790, 2000, v. 1.2.2.

DVB (Nov. 2004). Interaction Channel for Digital Terrestrial Television, ETSI EN 301 958, 2001, v. 1.1.1.

Giulietti A, Van der Perre L, and Strum A (Feb. 2002). Parallel turbo coding interleavers: Avoiding collisions in accesses to storage elements, Elec. Lett., **38**(5): 232 - 234.

Gross WJ and Gulak PG (Aug. 1998). Simplified MAP algorithm suitable for implementation of turbo decoder, IEEE Electronic Letter, **34**(16): 1577- 1578.

Hekstra A (1989). An alternative to metric rescaling in Viterbi decoders, IEEE Trans. Communication, **37**(11): 1220-1222.

Hagenauer J and Hoeher P (1989). A Viterbi algorithm with soft decision outputs and its applications. Communications Technology for the 1990s and Beyond, GLOBECOM'89, IEEE, **3**: 1680-1686.

IEEE, IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE 802.16-2004.

Jungu C and Ottosson T (May. 2000). Linearly approximated log-MAP algorithm for turbo decoding, Proceedure of Vehicular Technol. Conf. VTC'2000, Tokyo, Japan, 3, 2252-2256.

Matache A and Dolinar S (Aug. 2000). and Pollara F., Stopping rules for turbo decoders, in JPL TMO Progress Rep., **42**(142): 1-22.

Rovini M and Martinez A (Feb. 2006). Efficient stopping rule for turbo decoders, IEEE Electronics Letters, **42**(4): 235-236.

Third generation partnership project 2 (3GPP2). Physical Layer Standard for cdma2000 spread spectrum systems, Release D, Feb. 2004, 3GPP2 C.S0002-D, ver. 1.0.

Received: 16 August 2011; Revised : 07 December 2011; Accepted : 15 December 2011.