Low Complexity MAP Algorithm for Turbo Decoder

Jonghyun Seo and Jangmyung Lee

Dept. of Electronic Engineering, Pusan National University,
Jangjeon-dong, Geumjung-gu, Pusan, Korea, Republic of
{jhseo,jmlee}@pusan.ac.kr

Abstract. As a promising decoding algorithm for turbo codes in terms of relatively low BER, the maximum a posteriori (MAP) algorithm is most widely used. However, the conventional MAP algorithm requires a large number of computations. A modified MAP algorithm is therefore proposed for reduction of the associated memory size and ultimately power saving. A newly introduced block combing is performed for the memory efficiency such that two branch metrics (BMs) are merged into one branch metric. When calculating FSM (Forward State Metric) of the associated state transition, BM is included in the subsequent FSM, and thus when calculating APP (A Posteriori Probability), the BM is exempted and the number of computations for LLR (Log Likelihood Ratio) is reduced. Simulation results demonstrate reduced memory size in use and equivalent performance, compared to the conventional MAP algorithm.

Keywords: Turbo decoder, MAP algorithm.

1 Introduction

Error correcting code has been researched since announcement of Shannon‘s coding theorem in 1984. And turbo code is one of the error correcting codes proposed in 1993 by Berrou, Glavieux, and Thitimaishima [1].

Turbo code has been researched actively because it has excellent performance of error correction, and there are SOVA (Soft-Output Viterbi Algorithm) or MAP (Maximum A Posteriori) in algorithm of turbo code. Even though SOVA algorithm has less complexity than MAP algorithm, MAP algorithm shows better performance than SOVA algorithm [3]. Thus, MAP algorithm is mainly used in these days [4]. MAP algorithm proposed in 1974 by Bahl, and MAP algorithm calculates to APP from signal of noise.

The following section describes the MAP algorithm for the turbo decoder and then accounts to the block processing technique in section 3. Section 4 describes that efficient MAP algorithm using block combining. Section 5 describes experimental result. Finally section 6 summarizes this paper.

2 Map Algorithm

The MAP algorithm was firstly presented in 1974 by Bahl, Cocke, Jelinik and Raviv. The MAP algorithm aims to calculate the a-posteriori probability (APP) of each state transition [4][5].

Given noisy observation vector, the MAP algorithm finds the probability of each valid state transition as in the Trellis diagram. As shown in Fig.1, the terms, alpha, beta, and Gamma, are defined as the forward state metric and the backward state metric and the branch metric, respectively. Alpha has a systemic bit that means state metric transitioning from the previous state, s', at time k-1 to the next state, s, at time k. Beta can be obtained from the previous one by iterative calculation after receiving all information. The Gamma is defined as the probability that a given transition is chosen given the received sequence at a given state.[7]

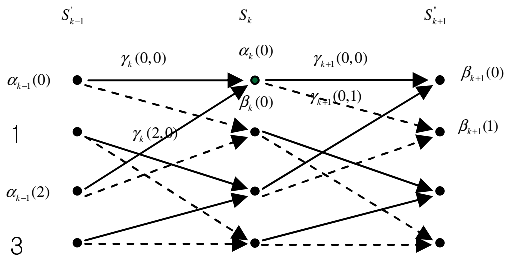

Fig. 1. The Conventional trellis with 4-state from time k-1 to time k+1

In order to calculate LLR, gamma value is initial needed from received data. Then, alpha and beta value are calculated by using gamma value. Numerical formula (Mathematic method) is shown below. Alpha and beta values are expressed as follows.

$$\alpha_k(s) = \sum_{all\ s'} \gamma_k(s',s) \times \alpha_{k-1}(s') \tag{1}$$

$$\beta_k(s') = \sum_{all\ s'} \beta_k(s',s) \times \gamma_k(s',s) \tag{2}$$

$$L(u_k) = \ln\left(\frac{\sum_{\substack{(s',s)=> \\ u_k=+1}} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{\substack{(s',s)=> \\ u_k=-1}} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right) \tag{3}$$

$$L(u_{k+1}) = \ln\left(\frac{\sum_{\substack{(s,s")=> \\ u_{k+1}=+1}} \alpha_{k-1}(s)\gamma_k(s,s")\beta_k(s")}{\sum_{\substack{(s,s")=> \\ u_{k+1}=-1}} \alpha_{k-1}(s)\gamma_k(s,s")\beta_k(s")}\right) \tag{4}$$

Eq. (3) and (4) are LLR computed at time k and k+1, respectively. These equations make a decision for the information bit with a maximum probability when transitioning from the previous state to the current state.

3 Using the Block Processing Technique

In this section, we explain a modified MAP algorithm using block processing technique for efficient memory use [2].

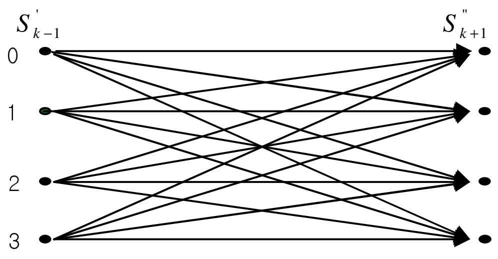

Fig. 2. Block processing trellis with 4-state from time k-1 to time k+1

The algorithm using the block processing technique does not take into account alpha and beta values at time k, but only at time k+1, thereby reducing memory storage for alpha and beta value at time k. Alpha, beta and gamma values are expressed as follows.

$$\alpha_{k+1}(s) = \sum_{all\ s'} \gamma(s',s,s'') \times \alpha_{k-1}(s')$$

$$\beta_{k-1}(s') = \sum_{all\ s''} \gamma(s'',s,s') \times \beta_{k-1}(s'')$$

$$\gamma(s'',s,s') = \gamma(s'',s) \times \beta_{k-1}(s'')$$

Although the block processing MAP algorithm needs much smaller memory size for the state metric and reduces power consumption, it has a defect that the algorithm needs more multiplication operations than conventional MAP algorithm in LLR calculation.

4 Proposed Scheme for Low Low Complexity Map Algorithm

A. The first proposed scheme

In this section, we explain a modified MAP algorithm using block combining for an efficient turbo decoder.

Fig. 3 shows a combining decoding process from two decoding processes. FSM and BSM are calculated using the combined BM values at each state, and the combined BM value means that two calculated BM values, one value is from time k-1 to k and the other is from time k to time k+1. The value can be expressed as,

$$\gamma_k^{s',s,s''} = \gamma_k^{s',s} \times \gamma_k^{s,s''} \tag{5}$$

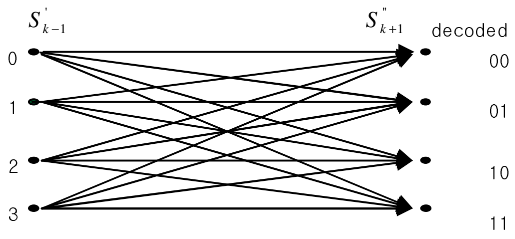

Fig. 3. Block combining trellis with 4-state from time k-1 to time k+1

$$\beta_{k-1}(s') = \sum_{all\ s''} \beta_k(s'') \times \gamma_k(s', s'') \tag{6}$$

$$\alpha_{k+1}(s) = \sum_{all\ s'} \gamma_k(s', s'') \times \alpha_{k-1}(s') \tag{7}$$

$\gamma_k^{s',s''}$ combining calculate value that transfer to state(S) in time k from state(S') in time k-1. Alpha and beta value in time k+1 calculate with integrated BM value Calculating Alpha and Beta value in this algorithm is as same as conventional MAP algorithm. That is, keep systemic bit to next state (s) at time k from previous state (s') at time k-1. Alpha in the same manner, received all of information bit thereafter current beta value repetitional calculate through next beta value.

To '0' state in time k+1 transfer from all states in time k-1 which receive data (0,0). To '1' state in time k+1 transfer from all states in time k-1 which receive data (0,1). And to '2' state in time k+1 transfer from time k-1 which receive data (1,0). To '3' state in time k+1 transfer from time k-1 which receive data (1,1). Probability values calculate that to each state in time k+1 from time k-1. Express followed below

$$p(u_k = 00 : y_k) = \sum_{0,0} \alpha_{k-1} \gamma_{s',s''} \beta_{k+1} \tag{8}$$

$$p(u_k = 01 : y_k) = \sum_{0,1} \alpha_{k-1} \gamma_{s',s''} \beta_{k+1} \tag{9}$$

$$p(u_k = 10 : y_k) = \sum_{1,0} \alpha_{k-1} \gamma_{s',s''} \beta_{k+1} \tag{10}$$

$$p(u_k = 11 : y_k) = \sum_{1,1} \alpha_{k-1} \gamma_{s',s''} \beta_{k+1} \tag{11}$$

$p(u_k = 00 : y_k)$ display probability likely data (0,0) in received data y_k. $p(u_k = 01 : y_k)$ indicates that probability likely data (0,1) in received data y_k.

And $p(u_k = 10 : y_k)$ indicates that probability likely (1,0). $p(u_k = 11 : y_k)$ indicates that probability likely (1,1).

After calculating their probability values, the maximum value can be determined because decoding calculation is differed according to the maximum value in LLRC. LLRC of the maximum value is as follows.

$$\text{CASE1 } p(u_k = 00 : y_k)$$
$$L(u_k) = \log(\frac{p(u_k = 10 : y_k)}{p(u_k = 00 : y_k)}) \ , \quad L(u_{k+1}) = \log(\frac{p(u_k = 01 : y_k)}{p(u_k = 00 : y_k)}) \tag{12}$$

$$\text{CASE2 } p(u_k = 01 : y_k)$$
$$L(u_k) = \log(\frac{p(u_k = 11 : y_k)}{p(u_k = 01 : y_k)}) \ , \quad L(u_{k+1}) = \log(\frac{p(u_k = 01 : y_k)}{p(u_k = 00 : y_k)}) \tag{13}$$

$$\text{CASE3 } p(u_k = 10 : y_k)$$
$$L(u_k) = \log(\frac{p(u_k = 10 : y_k)}{p(u_k = 00 : y_k)}) \ , \quad L(u_{k+1}) = \log(\frac{p(u_k = 11 : y_k)}{p(u_k = 10 : y_k)}) \tag{14}$$

$$\text{CASE4 } p(u_k = 11 : y_k)$$
$$L(u_k) = \log(\frac{p(u_k = 11 : y_k)}{p(u_k = 01 : y_k)}) \ , \quad L(u_{k+1}) = \log(\frac{p(u_k = 11 : y_k)}{p(u_k = 10 : y_k)}) \tag{15}$$

$L(u_k)$ become decoding value in time k. $L(u_{k+1})$ become decoding value in time k+1. $p(u_k = 00 : y_k)$ divide $p(u_k = 10 : y_k)$ in $L(u_k)$ of case1. It does divide $p(u_k = 00 : y_k)$ by $p(u_k = 10 : y_k)$ in $L(u_k)$ of case1. Because it does product of probability value of received data '0' when transfer to time k from time k-1 and probability value of received data '0' when transfer to time k+1 from time k. So, decoding value at time k+1 should make an offset.

Similarly, calculate $L(u_{k+1})$. Also case2 and case3 , case4 calculate in the same method. Equation (8) can be simplified.

$$p(u_k = 00 : y_k) = \sum_{0,0} \alpha_{k-1} \gamma_{s',s''} \beta_{k+1}$$
$$\begin{aligned} &= \alpha_{k-1}(0) \times \gamma_{0,0} \times \beta_{k+1}(0) + \alpha_{k-1}(1) \times \gamma_{1,0} \times \beta_{k+1}(0) + \alpha_{k-1}(2) \times \gamma_{2,0} \times \beta_{k+1}(0) + \alpha_{k-1}(3) \times \gamma_{3,0} \times \beta_{k+1}(0) \\ &= (\alpha_{k-1}(0) \times \gamma_{0,0} + \alpha_{k-1}(1) \times \gamma_{1,0} + \alpha_{k-1}(2) \times \gamma_{2,0} + \alpha_{k-1}(3) \times \gamma_{3,0}) \times \beta_{k+1}(0) \\ &= \alpha_{k+1}(0) \times \beta_{k+1}(0) \end{aligned} \tag{16}$$

Probability of (0,0) that is received data from equation (16) can be simplified by only current state of alpha and beta. Because alpha values at time k+1 include alpha values of all states. Equation (10), (11), (12) also can be simplified by same way.

B. The second proposed scheme

```
If $p(u_k = 00 : y_k)$ > $p(u_k = 01 : y_k)$ ;
 Then buf = ‘1’;
   If $p(u_k = 00 : y_k)$ > $p(u_k = 10 : y_k)$
     Decoding data = ‘00’;
   Else if $p(u_k = 00 : y_k)$ < $p(u_k = 10 : y_k)$
     Decoding data = ‘10’
Else if $p(u_k = 00 : y_k)$ < $p(u_k = 01 : y_k)$ ;
 Then buf = ‘0’;
  If $p(u_k = 01 : y_k)$ > $p(u_k = 11 : y_k)$;
   Decoding data = ‘01’;
  Else if $p(u_k = 01 : y_k)$ < $p(u_k = 11 : y_k)$;
   Decoding data = ‘11’;
```

<Pseudocode>

5 Conclusions

A Multiplication operation can be transferred to an adder operation by using Log-MAP algorithm[7]. To make an easy hardware design, Log-MAP algorithm applies to conventional algorithm and proposed algorithm. After that following the application, compare conventional algorithm with proposed algorithm into LLRC.

Table 1. Compare proposed Algorithm with conventional algorithm using log-MAP Algorithm in LLR Calculation(n:state number, m:data frame)

	Log-MAP algorithm	Pietroben algorithm	Block processing algorithm	Proposed algorithm
Adder Operation	4*n(m-1)	n(m-1)	12*n((m-1)/2)	1/2*n(m-1)
Max Operation	3/2*n(m-1)	1/2*n(m-1)	3*n(m-1)	(n-1)((m-1)/2)
FSM(or BSM) Memory	m*n	m*n	1/2*n(m+1)	1/2*n(m+1)
Number of total memory	2*n(2*m-1)	2*n(2*m-1)	n(3*m-1)	n(3*m-1)

Table 2. Compare Proposed Algorithm with conventional algorithm at the number of time for switching(using the Xilinx XST)

	Block processing algorithm	Proposed algorithm	The decrement(%)
Power(mw)	55.41	23.77	56
Area	70.08	28.68	59

Table 3. Compare Proposed Algorithm with conventional at LLRC(using the Xilinx XST)

	Conventional algorithm	Block processing algorithm & Proposed algorithm	The decrement(%)
n=4 m=400	6384	4796	25
n=4 m=800	12792	9596	25

The Proposed scheme has less adder operation than conventional algorithm, and memory uses are as same as block processing technique. However, the operation quantities are far less than block processing technique. Thus, turbo decoder design consist of proposed scheme, the total circuit area can be reduced because it uses small number of adders.

MAP algorithm of turbo decoder has an excellent performance of error correction, but it has very high complexity. In this paper, we proposed an efficient MAP algorithm by using the block combining. 'Block Combining' means that two times of decoding processes are unified with one process and the result both decrease the calculations in memory and operation.

Therefore, proposed MAP algorithm by using block combining can use in efficient memory size and in low power. It is also suitable for low power system or high-speed system.

References

1. Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon limit error-correcting coding and decoding: Turbo-Codes. In: Proceeding of IEEE International Conferences on Communications 1993, pp. 1064–1070 (May 1993)
2. Lee, I., Vallejo, M.L., Mujtaba, S.A.: Block Processing Technique for Low Power Turbo Decoder Design. In: IEEE 55th VTC Spring 2002, pp. 1025–1029 (2002)
3. Papke, L., Robertson, P., Villebrun, E.: Improved decoding with the SOVA in a parallel concatenated (trubo-code) scheme. In: Proceeding of ICC 1996, Dallas, TX, USA, pp. 102–106 (June 1996)
4. Bahl, L., Cocke, J., Jelinek, F., Raviv, J.: Optimal decoding of linear codes for minimizing symbol error rate. IEEE Trans. Inform. Theory IT-20, 284–287 (1974)

5. Pietrobon, S., Barbulescu, A.S.: A simplification of the modified Bahl decoding algorithm for systematic convolutional codes. In: Proceeding of ISITA 1994, Sydney, Australia, pp. 875–880 (November 1994)
6. Pietrobon, S., Barbulescu, A.S.: A simplification of the modified Bahl decoding algorithm for systematic convolutional codes. In: Proceeding of ISITA 1994, Sydney, Australia, pp. 875–880 (November 1994)
7. Robertson, P., Villebrun, E., Hoher, P.: A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain. In: Proceedings of the International Conference on Communications, pp. 1009–1013 (June 1995)
8. Forney, G.: The Viterbi algorithm. Processdings of the IEEE 61, 268–278 (1973)
9. Bahl, L.R., Cocke, J., Jelink, F., Raviv, J.: Optimal Decoding of Linear Codes for Minimising Symbol Error Rate. IEEE Transactions on Information Theory 20, 284–287 (1974)
10. Shrestha, R., Paily, R.: 2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), pp. 86–91 (2013)