Efficient Method to Implement Max-Log-MAP Algorithm: Parallel SOVA

Chang-Woo Lee* ***Regular Member***

ABSTRACT

The efficient method to implement the Max-Log-MAP algorithm is proposed by modifying the conventional algorithm. It is called a parallel soft output Viterbi algorithm (SOVA) and the rigorous proof is given for the equivalence between the Max-Log-MAP algorithm and the parallel SOVA. The parallel SOVA is compared with the conventional algorithms and we show that it is an efficient algorithm implementing the modified SOVA in parallel.

Key Words : Parallel SOVA, Max-Log-MAP algorithm, Iterative codes, Turbo codes, Efficient method

Ⅰ. Introduction

In order to reduce the computational complexity of the maximum a posteriori (MAP) algorithm, which is the optimum solution to decode iterative codes such as turbo codes, the Log-MAP algorithm is proposed[1,2]. In the Log-MAP algorithm, by transferring its operations to the log domain, multiplications are replaced by additions and exponential calculations are not required. The complexity can be much reduced by approximating the Log-MAP algorithm with a small performance degradation in the Max-Log-MAP algorithm[2]. The efficient algorithm to implement the Max-Log-MAP algorithm has been proposed in [3].

In this paper, we modify the efficient algorithm proposed in [3], and it will be called as a parallel SOVA, since it can be viewed as a parallel implementation of the modified SOVA. Then, the rigorous proof is given for the equivalence between the parallel SOVA and the Max-Log-MAP algorithm. The equivalence between the parallel SOVA and the Max-Log-MAP algorithm is proved by induction. Then, the parallel SOVA is compared with the conventional algorithms and we show that it is an efficient algorithm implementing the modified SOVA in parallel.

This paper is organized as follows. The conventional decoding algorithms for iterative codes are given in Section 2. In Section 3, the parallel SOVA is presented. The equivalence between the parallel SOVA and the Max-Log-MAP algorithm is proved and the parallel SOVA is compared with the conventional algorithms in Section 4. Finally, concluding remarks are given in Section 5.

Ⅱ. Decoding algorithms for iterative codes

The MAP algorithm is optimal component decoder for turbo codes, and it can be summarized as follows[2,4,5,6]. If u_k is the k-th decoded bit and y is the received symbol sequence, the log-likelihood ratio (LLR), which is used as a measure of reliability, is given by

※ This study was supported by the Research Fund, 2007-2 of The Catholic University of Korea.

* School of Information, Communications and Electronics Engineering, The Catholic University of Korea (changwoo@catholic.ac.kr)

논문번호 : KICS2007-11-496, 접수일자 : 2007년 11월 6일, 최종논문접수일자 : 2008년 6월 20일

$$L(u_k) = \ln\left(\frac{P(u_k=+1|y)}{P(u_k=-1|y)}\right) \quad (1)$$

$$= \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1} \alpha_{k-1}(s') \cdot \gamma_k(s',s) \cdot \beta_k(s)}{\sum_{(s',s)\Rightarrow u_k=-1} \alpha_{k-1}(s') \cdot \gamma_k(s',s) \cdot \beta_k(s)}\right)$$

where s' and s are the starting and ending states of each stage in the trellis, respectively. The $\alpha_k(s')$ and $\beta_k(s)$ are the forward and backward metrics, which are calculated by using the forward recursion and backward recursion, respectively. The $\gamma_k(s',s)$ is the branch metric. However, the multiplications used in the recursive calculation of the $\alpha_k(s')$ and $\beta_k(s)$, and the exponents used to calculate the $\gamma_k(s',s)$ terms make the MAP algorithm extremely complex.

The Log-MAP algorithm is theoretically identical to the MAP algorithm, while the associated complexity is significantly reduced by transferring its operation to the log domain. In the Log-MAP algorithm, the logarithmic value is calculated by

$$\ln P(u_k = a|y) \quad (2)$$

$$= M^*_{(s',s)\Rightarrow u_k=a}(A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)),$$

where a is *+1* or *-1*. $A_k(s')$, $\Gamma_k(s',s)$ and $B_k(s)$ are the log values of $\alpha_k(s')$, $\gamma_k(s',s)$and $\beta_k(s)$, respectively. $M^*_{cond.}(X_i)$ can be calculated by using the following Jacobian logarithm repetitively,

$$M^{*2}(X_m, X_n) \equiv \ln(e^{X_m} + e^{X_n}) \quad (3)$$

$$= \max(X_m, X_n) + \ln(1 + e^{-|X_m - X_n|}),$$

where X_m and X_n are two $X_i's$ satisfying *cond.* $A_k(s)$ and $B_k(s)$ are computed using the following forward and backward recursions for each state at each stage,

$$A_k(s) = M^{*2}((A_{k-1}(s') + \Gamma_k(s',s))_{(s',s)\Rightarrow u_k=+1}, \quad (4)$$

$$(A_{k-1}(s') + \Gamma_k(s',s))_{(s',s)\Rightarrow u_k=-1}), \quad k = 1, \cdots, K$$

and

$$B_{k-1}(s) = M^{*2}((\Gamma_k(s',s) + (B_k(s))_{(s',s)\Rightarrow u_k=+1}, \quad (5)$$

$$(\Gamma_k(s',s) + (B_k(s))_{(s',s)\Rightarrow u_k=-1}), \quad k = 1, \cdots, K$$

In order to calculate the log-likelihood ratio, all branches for each stage in the trellis are divided into two classes satisfying $u_k = +1$ and $u_k = -1$, respectively, and $M^*_{(s',s)\Rightarrow u_k=a}(A_{k-1}(s') + \Gamma_k(s',s) + B_k(s))$ in formula (2) is then computed for each group using formula (3) repetitively. By omitting the log term in formula (3), we can make an approximation to the Log-MAP algorithm, which is called the Max-Log-MAP algorithm. The Max-Log-MAP algorithm finds the LLR $L(u_k|y)$ for a given bit u_k by comparing the probability of the most likely path giving $u_k = +1$ with the probability of the most likelihood path giving $u_k = -1$.

In the SOVA, the log-likelihood ratio is calculated by taking the metric difference, $\Delta_i(S_i)$, as the reliability value. The LLR value at time *k*, $L(u_k|y)$, is calculated using formula[2,7]

$$L(u_k|y) = sign(u_k) \cdot \min_{i=k,\cdots,k+\delta, u_i \neq u_k^i} \Delta_i(S_i) \quad (6)$$

where u_k is the value of the bit given by the ML path, and u_k^i is the value of this bit for the path which merged with the ML path and is discarded at trellis stage *i*, as is shown in Fig. 1. The SOVA gives a deraded performance compared to the Max-Log-MAP algorithm, as is explained in Section 4. By considering not only the competing path but also those paths which merge into the competing path, such as path-n in Fig. 1 in LLR updates, the modified SOVA has the same performance as the Max-Log-MAP algorithm[7]. In the modified SOVA, if $u_k^{k+3} = u_k$ and $u_k'' \neq u_k$, the LLR is updated by

$$L(u_k|y) = sign(u_k) \cdot \min(L_k, L_j'' + \Delta_{k+3}(S_0)), \quad (7)$$

where L_j'' represents the reliability difference between the path-2 and the path-n and L_k is the reliability value calculated in the SOVA.

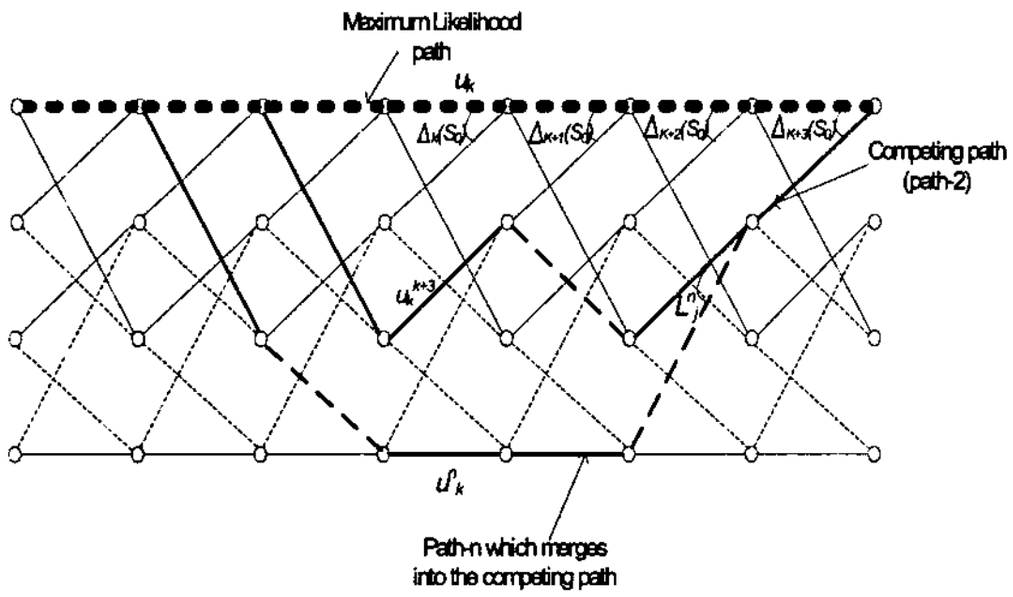

Fig. 1. Trellis description with respect to the reliability

Ⅲ. Parallel SOVA

The performance of iterative decoders generally improves until the number of iterations reaches a certain value. Thus, it is essential to reduce the computational complexity of the decoding algorithm. An efficient algorithm to decode iterative codes has been proposed in [3], and a modified version of the algorithm, which is called as a parallel SOVA, is presented in this Section. In the next Section, a rigorous proof for the equivalence between the parallel SOVA and the conventional Max-Log-MAP algorithm is given. The parallel SOVA is described as follows.

The forward recursion of the parallel SOVA is the same as that of the conventional Max-Log-MAP algorithm, except for the storage of the difference metric. The difference metric, $\Delta_k(s_m)$, is saved as

$$\Delta_k(s_m) = |(A_{k-1}(s_i') + \Gamma_k(s_i', s_m))_{u_k=+1} - (A_{k-1}(s_j') + \Gamma_k(s_j', s_m))_{u_k=-1}|. \quad (8)$$

In the backward recursion, the path metric, $D_k^{cand.}(s_m)$, is assigned to each branch. It is the accumulated value of the difference metrics, as given by

$$D_k^{u_k=a}(s_m) = \begin{cases} (B_k'(s_m))_{u_k=a}, \\ \text{if } \textit{the branch is selected at forward recursion at state } s_m \textit{ of stage } k \\ (B_k'(s_m) + \Delta_k(s_m))_{u_k=a}, \\ \text{if } \textit{the branch is not selected at forward recursion at state } s_m \textit{ of stage } k \end{cases} \quad (9)$$

where the new backward metric, $B_k'(s_i)$, is the minimum value of the path metrics in the backward recursion, as given by

$$B_{k-1}'(s_i') = \min(D_k^{u_k=-1}(s_l), D_k^{u_k=+1}(s_m)). \quad (10)$$

The new backward recursion of the parallel SOVA is illustrated in Fig. 2. This value is accumulated with the minimum differences between the ML path and the path including the corresponding branch, as can be seen in formulas (9) and (10).

The value of the path metric represents the difference of reliability between the maximum likelihood (ML) path and the path which includes the specific branch at stage k. The value of the path metric for the ML path is zero, while those for the other paths are always greater than zero. The initial value of $B_k'(s_i)$ is zero or infinity if the trellis is terminated. If the trellis is not terminated, the value of $B_k'(s_i)$ for the state on the estimated ML path can be set to zero. The LLR is calculated directly as

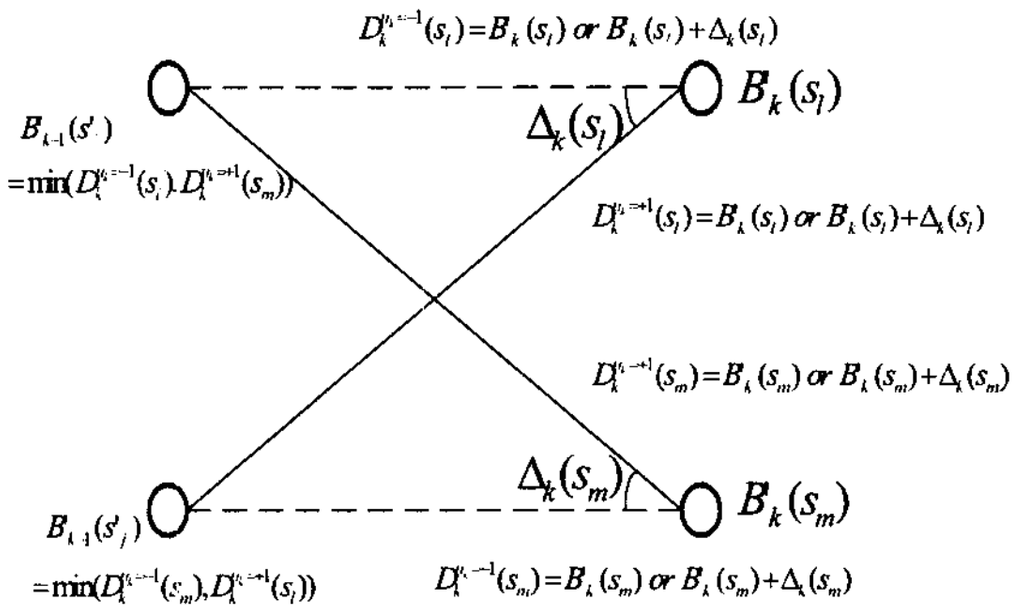

Fig. 2. Backward recursion for the proposed algorithm

$$L(u_k|\underline{y}) = \min_{m'}(D_k^{u_k=-1}(s_{m'})) - \min_{n'}(D_k^{u_k=+1}(s_{n'})) \quad (11)$$

Notice that the minimum value of the path metric in formula (11) is the minimum value among the relative differences of reliability between the ML path and the path which includes the branch satisfying each condition at stage k.

The number of operations involving real numbers can be analyzed to compare the complexities between the parallel SOVA and the conventional Max-Log-MAP algorithm. The forward recursion for the parallel SOVA is same as that of the conventional Max-Log-MAP algorithm, except for the saving of the difference metric. In the backward recursion, the number of additions of the parallel SOVA is half of that for the Max-Log-MAP algorithm, since additions are needed for only a half of the paths. In calculating the LLR, two additions in formula (2) of the conventional algorithm are no longer needed in the parallel SOVA. Moreover, if the path is the ML path, then the minimum value for the path metric, $\min(D_k^{cond.}(s'))$, is known to be zero, without the need of calculation, when using the parallel SOVA. The number of additions required by the parallel SOVA is less than 40% of that required by the conventional Max-Log-MAP algorithm, as was analyzed in [3].

Ⅳ. Comparison between the parallel SOVA and the conventional algorithms

4.1 Equivalence between the parallel SOVA and the Max-Log-MAP algorithm

The parallel SOVA produces the same LLR value as the conventional Max-Log-MAP algorithm, and the rigorous proof for that is given as follows. First, in order to illustrate the backward recursion of the parallel SOVA, the path metrics for the last two stages are shown in Fig. 3. For the sake of simplicity, we assume that the ML path consists of all zero bits. The path metrics for the two branches at stage K are zero and $(A_{K-1}(s_0')+\Gamma_K(s_0',s_0))-(A_{K-1}(s_1')+\Gamma_K(s_1',s_0))$, respectively. Thus, the backward metric of the state s_n at stage *K-1* is given by

$$B_{K-1}'(s_n') = (A_{K-1}(s_0')+\Gamma_K(s_0',s_0)) - (A_{K-1}(s_n')+\Gamma_K(s_n',s_0)). \quad (12)$$

In order to prove the equivalence between the parallel SOVA and the Max-Log-MAP algorithm by induction, let's assume that the backward metric of the state s_n at an arbitrary stage k is given by

$$B_k'(s_n') = (A_k(s_0')+H_k(s_0')) - (A_k(s_n')+H_k(s_n')). \quad (13)$$

Then, the path metric can be calculated as follows. First, let's assume that the branch, which enters the state s_n at stage k from the state s_i at stage *k-1*, is selected at the state s_n at stage k. Then, the path metric, $D_k^{u_k=a}(s_n)$, for the selected branch is equal to the backward metric, $B_k'(s_n)$, by formula (9). In this case, since $A_k(s_n)$ is equal to $A_{k-1}(s_i')+\Gamma_k(s_i',s_n)$ in the forward recursion, the path metric is calculated as

$$\begin{aligned} D_k^{u_k=a}(s_n) &= (A_k(s_0)+H_k(s_0')) - (A_k(s_n)+H_k(s_n')) \quad (14)\\ &= (A_{k-1}(s_0')+\Gamma_k(s_0',s_0)+H_k(s_0')) \\ &\quad -(A_{k-1}(s_i')+\Gamma_k(s_i',s_n)+H_k(s_n')) \end{aligned}$$

On the other hand, if the branch, which enters the state s_n at stage k from the state s_i at stage *k-1*, is not selected, the path metric for the branch is equal to $B_k'(s_n)+\Delta_k(s_n)$ by formula (9). Since the value of $A_k(s_n)$ is equal to that of $A_{k-1}(s_j')+\Gamma_k(s_j',s_n)$, the path metric can be calculated as

$$\begin{aligned} D_k^{u_k=a}(s_n) &= (A_k(s_0)+H_k(s_0')) - (A_k(s_n)+H_k(s_n')) \\ &\quad +(A_{k-1}(s_j')+\Gamma_k(s_j',s_n)) - (A_{k-1}(s_i')+\Gamma_k(s_i',s_n)) \\ &= (A_{k-1}(s_0')+\Gamma_k(s_0',s_0)+H_k(s_0')) \\ &\quad -(A_{k-1}(s_i')+\Gamma_k(s_i',s_n)+H_k(s_n')) \end{aligned} \quad (15)$$

which is the same result as that obtained using formula (14). Then, the backward metric of the state i at the stage *k-1*($k<K$) is given by

$$B_{k-1}'(s_i') = (A_{k-1}(s_0')+\Gamma_k(s_0',s_0)+H_k(s_0')) - (A_{k-1}(s_i')+\Gamma_k(s_i',s_n)+H_k(s_n')), \quad (16)$$

where the backward metric is the minimum value between the two path metrics at each state of each

stage. Since $H_k(s_i')$ is equal to $\Gamma_K(s_i', s_0)$ at stage *K-1* in (13), we can show from formulas (12), (13) and (16), by induction, that the backward metric at the state s_n of the stage $k(0 \le k \le K-1)$ is given by

$$B_k'(s_n') = (A_k(s_0') + \sum_{l=k+1}^{K} \Gamma_l(s_0', s_0)) \quad (17)$$

$$-(A_k(s_n') + \sum_{l=k+1}^{K} \Gamma_l(s_{l_{st1}}', s_{l_{st2}})),$$

where $s'_{l_{st1}}$ and $s_{l_{st2}}$ are the starting and ending states at each stage, respectively. The path metric, $D_k^{u_k=a}(s_n)$, is given by

$$D_k^{u_k=a}(s_n) = (A_{k-1}(s_0') + \Gamma_k(s_0', s_0) + \sum_{l=k+1}^{K} \Gamma_l(s_0', s_0))$$

$$-(A_{k-1}(s_i') + \Gamma_k(s_i', s_n) + \sum_{l=k+1}^{K} \Gamma_l(s'_{l_{st1}}, s_{l_{st2}})). \quad (18)$$

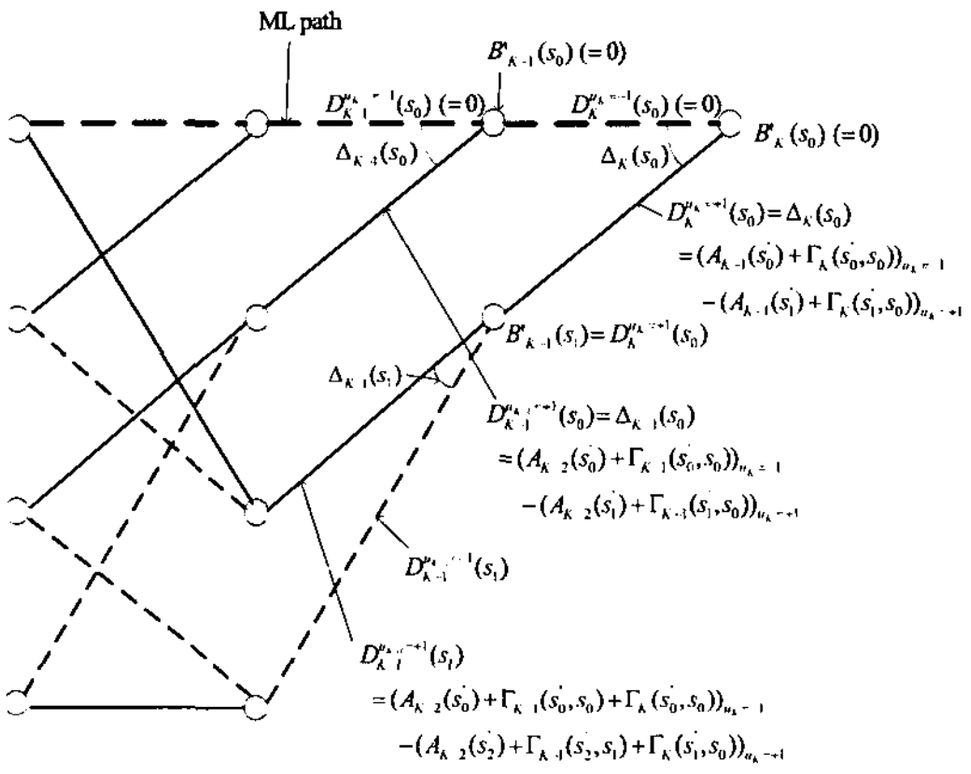

Fig. 3. Metrics for the last two stage

Since the backward metric is accumulated with the minimum path metric and the path metric represents the difference of reliability between the ML path and the path which includes the branch, the backward metric represents the minimum difference of reliability between the ML path and the path which includes the corresponding branch. Thus, the LLR value calculated in (11) is the same as that calculated by the conventional Max-Log-MAP algorithm.

4.2 Comparing various decoding algorithms

The decoding algorithms for iterative codes can be compared as follows. First, the SOVA gives a degraded performance compared to the Max-Log-MAP algorithm due to the following reasons[2]. In the Max-Log-MAP algorithm, once the path merges with the ML path, it will have the same value of $B_k(s)$ as the ML path. Hence, taking the difference between the metrics of the two merging paths in the SOVA is equivalent to taking the difference between two values of $(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s))$ in the Max-Log-MAP algorithm. The only difference is that in the Max-Log-MAP algorithm one path will be the ML path, and the other will be the most likely path that gives a different hard decision for u_k. On the other hand, in the SOVA one path will be the ML path, but the other will be the most likely path that gives a different hard decision for u_k and survives to merge with the ML path. More likely paths, which give a different hard decision for the bit u_k may have been discarded before they merge with the ML path.

In the parallel SOVA, the reliability difference is calculated using the minimum path metrics in formula (11), where one of the minimum value is zero for the ML path and the other is the minimum value among the reliability differences between the ML path and the pathes which include branches with the different bit value from the ML path. Those pathes are either the competing pathes merging with the ML path or the pathes that eventually merge with the competing path. Thus, the parallel SOVA provides the same performance with the modified SOVA. The parallel SOVA can be viewed as the parallel implementation of the modified SOVA, since the metric difference and the accumulated difference are calculated in parallel at each state of each stage. It can be implemented more regularly than the modified SOVA.

V. Conclusion

The rigorous proof for the equivalence between the parallel SOVA and the Max-Log-MAP algorithm was given and the various decoding algorithms were compared. It can be concluded that the parallel SOVA is an efficient algorithm to implement the Max-Log-MAP algorithm and it can be also considered as the parallel implementation of the modified SOVA. Thus, the parallel SOVA can be used as an efficient method to decode iterative codes such as turbo codes.

References

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," IEEE Trans. Commun., Vol.44, No.10, pp.1261-1271, Oct. 1996.

[2] J.P. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: an overview," IEEE Trans. Veh. Technol., Vol.49, No.6, pp.2208-2233, June 2000.

[3] C. W. Lee, "Efficient algorithm for decoding concatenated codes," IEICE Trans. on Commun., Vol.E87-B, No.11, pp.3180-3187, Nov. 2004.

[4] W.J. Gross and P.G.Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," Electronics Letters, Vol.34, No.16, pp.1577-1578, 1998.

[5] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. Inform. Theory, Vol.20, No.2, pp.284-287, Mar. 1974.

[6] L. Hanzo, T.H. Liew and B.L. Yeap, Turbo coding, turbo equalization and space-time coding, John Wiley and Sons, 2002.

[7] M.P.C. Fossorier, F. Burkert, S. Lin and J. Hagenauer, "On the equivalence between SOVA and Max-Log-MAP decodings," IEEE Commun. Letters, Vol.2 No.5, pp.137-139, May 1998.

이 창 우(Chang-Woo Lee) 정회원

현 재 가톨릭대학교 정보통신전자공학부 교수

<관심분야> 영상 통신, 영상 압축, turbo 부호