Matrix Approach for Fast Implementations of Logarithmic MAP Decoding of Turbo Codes

Duanyi Wang and Hisashi Kobayashi

Department of Electrical Engineering, Princeton University
Princeton, NJ 08544, USA
Tel: (609) 258-4643 Fax: (609) 258-2158
{duanyi, hisashi}@ee.princeton.edu

Abstract

This paper describes two new matrix transform algorithms for the Max-Log-MAP decoding of turbo codes. In the proposed algorithms, the successive decoding procedures carried out in the conventional Max-Log-MAP algorithm are performed in parallel, and well formulated into a set of simple and regular matrix operations, which can therefore considerably speed up the decoding operations and reduce the computational complexity. The matrix Max-Log-MAP algorithms also maintain the advantage of the general logarithmic MAP like algorithms in avoiding complex numerical representation problems. They particularly facilitate the implementations of the logarithmic MAP like algorithms in special-purpose parallel processing VLSI hardware architectures. The matrix algorithms also allow simple implementations by using shift registers. The proposed implementation architectures for the matrix Max-Log-MAP decoding can effectively reduce the memory capacity and simplify the data accesses and transfers required by the conventional Max-Log-MAP as well as MAP algorithms.

Keywords: Turbo codes, MAP algorithm, Max-Log-MAP algorithm, Matrix transform, VLSI.

1. Introduction

Maximum *a posteriori* probability (MAP) decoding, based on the BCJR algorithm [1], has seen a resurgence of interest since it was well used for the iterative decoding of turbo codes [2]. Its implementation in VLSI circuits is increasingly in demand in recent years to apply turbo codes to a number of practical systems. However, the original MAP algorithm suffers from serious drawbacks in its implementation. The main technical difficulty is its complex numerical representation problem, basically due to non-linear functions and a large number of multiplications and additions involved. To overcome this disadvantage, Max-Log-MAP and Log-MAP algorithms have been proposed [3]. In both algorithms, the processing is done exclusively in the logarithmic domain, and the operations involved (only "add" and "maximum" function) are easier to handle. Up to now, almost all of the practical MAP decoders have been implemented in such logarithmic MAP-like algorithms.

In this paper, we present two matrix transform algorithms for the Max-Log-MAP decoding of turbo codes. In the proposed algorithms, the successive decoding procedures carried out in the conventional Max-Log-MAP algorithm are performed in parallel, and well formulated into a set of simple and regular matrix operations, which can therefore considerably speed up the decoding operations and reduce the computational complexity. Our matrix approaches also maintain the advantage of the general logarithmic algorithms in avoiding the complex numerical representation problem. They particularly facilitate the implementation of the Max-Log-MAP algorithm in special-purpose VLSI hardware for parallel processing.

2. Principle of Matrix Max-Log-MAP Decoding

2.1 System Model and Max-Log-MAP Algorithm

Consider the transmission system of Figure 1. The turbo encoder is constructed by parallel concatenation of two RSC (recursive systematic convolutional) codes and an interleaver in-between (Figure 2). For each input information sequence block $\boldsymbol{d}_1^N = \{d_1, d_2, \ldots, d_N\}$, where $d_k \in GF(2)$, the two RSC encoders operate it on different versions and produce the parity sequences $\boldsymbol{Y}_1 = \{Y_{11}, Y_{12}, \ldots, Y_{1N}\}$ and $\boldsymbol{Y}_2 = \{Y_{21}, Y_{22}, \ldots, Y_{2N}\}$. So the overall turbo encoded output sequence is $\boldsymbol{C}_1^N = \{C_1, C_2, \ldots, C_N\}$, where $C_k = (X_k, Y_{1k}, Y_{2k})$. We may write simply $\boldsymbol{C}_1^N = (\boldsymbol{X}, \boldsymbol{Y}_1, \boldsymbol{Y}_2)$, The encoded sequence is then sent over the channel. At the receiver side, the input sequence to the turbo decoder is denoted $\boldsymbol{R}_1^N = \{R_1, R_2, \ldots, R_N\}$, where $R_k = (x_k, y_{1k}, y_{2k})$ is the noise corrupted version of C_k at time k (assume sufficient channel interleaving).

Let the state of the RSC encoder at time k be S_k, which

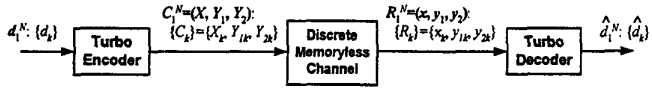


Figure 1 Transmission system model

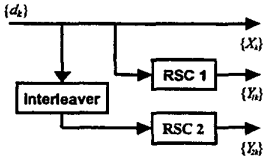


Figure 2
Turbo encoder

takes on an integer value between 0 and M-1 ($M = 2^v$, v is the memory of RSC encoder). The k-th information bit d_k drives the encoder to change its state from S_{k-1} to S_k. When the Max-Log-MAP algorithm is adopted, the soft output for each decoded bit d_k is determined as follows:

$$\Lambda(d_k) \approx \max_{(S_k, S_{k-1})} \left(\bar{\gamma}_1(R_k, S_{k-1}, S_k) + \alpha_{k-1}(S_{k-1}) + \bar{\beta}_k(S_k)\right) - \max_{(S_k, S_{k-1})} \left(\bar{\gamma}_0(R_k, S_{k-1}, S_k) + \alpha_{k-1}(S_{k-1}) + \bar{\beta}_k(S_k)\right) \quad (1)$$

where $\bar{\alpha}_k(S_k)$ and $\bar{\beta}_k(S_k)$ are recursively obtainable by

$$\bar{\alpha}_k(S_k) \approx \max_{(S_{k-1}, i)} \left(\bar{\gamma}_i(R_k, S_{k-1}, S_k) + \bar{\alpha}_{k-1}(S_{k-1})\right) \quad (2)$$

$$\bar{\beta}_k(S_k) \approx \max_{(S_{k+1}, i)} \left(\bar{\gamma}_i(R_k, S_k, S_{k+1}) + \bar{\beta}_{k+1}(S_{k+1})\right) \quad (3)$$

and S_{k-1}, S_k, $S_{k+1} = 0, 1, \ldots, M\text{-}1$, $i = 0, 1$.

(Note the $\bar{\gamma}_i(R_k, S_{k-1}, S_k)$, $\bar{\alpha}_k(S_k)$ and $\bar{\beta}_k(S_k)$ are defined by

$$\bar{\gamma}_i(R_k, S_{k-1}, S_k) = \ln \gamma_i(R_k, S_{k-1}, S_k),$$
$$\bar{\alpha}_k(S_k) = \ln \alpha_k(S_k), \quad \bar{\beta}_k(S_k) = \ln \beta_k(S_k)$$

and $\gamma_i(R_k, S_{k-1}, S_k)$ is the branch transition probability.

2.2 Matrix Algorithm for Max-Log-MAP Decoding

Define an $M \times M$ matrix

$$\bar{\Gamma}_i(R_k) = \begin{bmatrix} \bar{\gamma}_i(R_k, 0, 0) & \bar{\gamma}_i(R_k, 0, 1) & \cdots & \bar{\gamma}_i(R_k, 0, M\text{-}1) \\ \bar{\gamma}_i(R_k, 1, 0) & \bar{\gamma}_i(R_k, 1, 1) & \cdots & \bar{\gamma}_i(R_k, 1, M\text{-}1) \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\gamma}_i(R_k, M\text{-}1, 0) & \bar{\gamma}_i(R_k, M\text{-}1, 1) & \cdots & \bar{\gamma}_i(R_k, M\text{-}1, M\text{-}1) \end{bmatrix} \quad (4)$$

In a similar way, define matrix

$$\bar{\Gamma}(R_k) = \begin{bmatrix} \bar{\gamma}(R_k, 0, 0) & \bar{\gamma}(R_k, 0, 1) & \cdots & \bar{\gamma}(R_k, 0, M\text{-}1) \\ \bar{\gamma}(R_k, 1, 0) & \bar{\gamma}(R_k, 1, 1) & \cdots & \bar{\gamma}(R_k, 1, M\text{-}1) \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\gamma}(R_k, M\text{-}1, 0) & \bar{\gamma}(R_k, M\text{-}1, 1) & \cdots & \bar{\gamma}(R_k, M\text{-}1, M\text{-}1) \end{bmatrix} \quad (5)$$

where

$$\bar{\gamma}(R_k, S_{k-1}, S_k) = \left\{\bar{\gamma}_0(R_k, S_{k-1}, S_k),\ \bar{\gamma}_1(R_k, S_{k-1}, S_k)\right\} \quad (6)$$

and vectors

$$\bar{\boldsymbol{\alpha}}_k = \left[\bar{\alpha}_k(0), \bar{\alpha}_k(1), \cdots, \bar{\alpha}_k(M\text{-}1)\right] \quad (7)$$
$$\bar{\boldsymbol{\beta}}_k = \left[\bar{\beta}_k(0), \bar{\beta}_k(1), \cdots, \bar{\beta}_k(M\text{-}1)\right] . \quad (8)$$

So, (1)-(3) become

$$\bar{\boldsymbol{\alpha}}_k \approx \bar{\boldsymbol{\alpha}}_{k-1} \nabla \bar{\boldsymbol{\Gamma}}(R_k) \qquad k = 1, 2, \ldots, N\text{-}1 \quad (9)$$
$$\bar{\boldsymbol{\beta}}_k^{\mathrm{T}} \approx \bar{\boldsymbol{\Gamma}}(R_{k+1}) \nabla \bar{\boldsymbol{\beta}}_{k+1}^{\mathrm{T}} \qquad k = N\text{-}1, N\text{-}2, \ldots, 1 , \quad (10)$$

where the operation ∇ is defined as in (2) and (3). And $\Lambda(d_k)$ becomes

$$\Lambda(d_k) \approx \boldsymbol{\alpha}_{k-1} \nabla' \boldsymbol{\Gamma}_1(R_k) \nabla' \boldsymbol{\beta}_k^{\mathrm{T}} - \boldsymbol{\alpha}_{k-1} \nabla' \boldsymbol{\Gamma}_0(R_k) \nabla' \boldsymbol{\beta}_k^{\mathrm{T}} \quad (11)$$

where the operation ∇' means:

$$\bar{\alpha}_k(S_k) \approx \max_{(S_{k-1})} \left(\bar{\gamma}_i(R_k, S_{k-1}, S_k) + \bar{\alpha}_{k-1}(S_{k-1})\right) \quad (12)$$
$$\bar{\beta}_k(S_k) \approx \max_{(S_{k+1})} \left(\bar{\gamma}_i(R_k, S_k, S_{k+1}) + \bar{\beta}_{k+1}(S_{k+1})\right) , \quad (13)$$

where S_{k-1}, $S_k = 0, 1, \ldots, M\text{-}1$, $i = 0, 1$.

From (9) and (10), we can obtain immediately

$$\bar{\boldsymbol{\alpha}}_k \approx \bar{\boldsymbol{\alpha}}_0 \nabla \bar{\boldsymbol{\Gamma}}(R_1) \nabla \bar{\boldsymbol{\Gamma}}(R_2) \nabla \cdots \nabla \bar{\boldsymbol{\Gamma}}(R_k), \quad k=1,2,\ldots,N\text{-}1 \quad (14)$$
$$\bar{\boldsymbol{\beta}}_k^{\mathrm{T}} \approx \bar{\boldsymbol{\Gamma}}(R_{k+1}) \nabla \bar{\boldsymbol{\Gamma}}(R_{k+2}) \nabla \cdots \nabla \bar{\boldsymbol{\Gamma}}(R_N) \nabla \boldsymbol{\beta}_N^{\mathrm{T}}, \quad k=N\text{-}1,N\text{-}2,\ldots,1 \quad (15)$$

So we have the matrix algorithm for Max-Log-MAP decoding as follows:

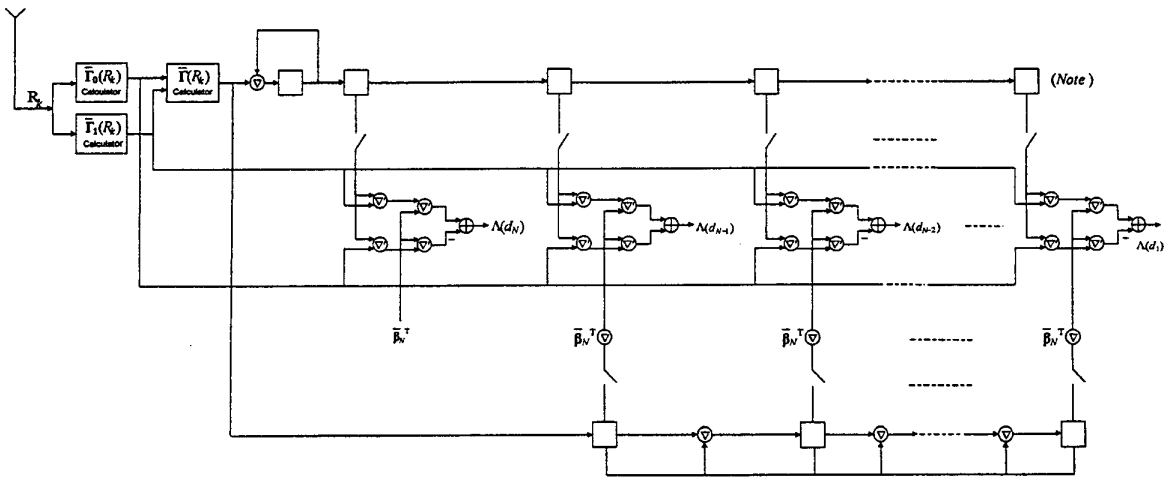
Matrix Max-Log-MAP Algorithm 1

1) Initialize $\bar{\boldsymbol{\alpha}}_0$ and $\bar{\boldsymbol{\beta}}_N$ by
$\bar{\boldsymbol{\alpha}}_0 = [0, *, *, \cdots, *]$ and $\bar{\boldsymbol{\beta}}_N = [0, *, \cdots, *]$, where $0 < * < -\infty$.
2) For each observation R_k, $\bar{\boldsymbol{\Gamma}}_i(R_k)$ and $\bar{\boldsymbol{\Gamma}}(R_k)$ are computed using (4) and (5), respectively. Then compute $\bar{\boldsymbol{\alpha}}_k$ using (9), and at the same time, compute $\bar{\boldsymbol{\Gamma}}(R_2) \nabla \bar{\boldsymbol{\Gamma}}(R_3) \nabla \cdots \nabla \bar{\boldsymbol{\Gamma}}(R_k)$, $\bar{\boldsymbol{\Gamma}}(R_3) \nabla \bar{\boldsymbol{\Gamma}}(R_4) \nabla \cdots \nabla \bar{\boldsymbol{\Gamma}}(R_k)$, ..., $\bar{\boldsymbol{\Gamma}}(R_{k-1}) \nabla \bar{\boldsymbol{\Gamma}}(R_k)$, in parallel.
3) After receiving the complete sequence $\boldsymbol{R}_1^N$, compute $\bar{\boldsymbol{\beta}}_k^{\mathrm{T}}$, $k = 1, 2, \ldots, N\text{-}1$, using (15) in parallel. Then compute $\Lambda(d_k)$, $k = 1, 2, \ldots, N$, using (11).

Its implementation architecture is shown in Figure 3.

According to Eqs. (14) and (15), we can further have

Theorem: Suppose $\bar{\boldsymbol{\Gamma}}(R_k)$, $\bar{\boldsymbol{\Gamma}}_i(R_k)$, $\bar{\boldsymbol{\alpha}}_k$, $\bar{\boldsymbol{\beta}}_k$ and $\Lambda(d_k)$ are defined by (4-8) and (11), then



Note: The shift register with length of $N+1$ is initialized by (α_0 , #, ... , #), where $-\infty < \# < +\infty$.

Figure 3 Implementation architecture of the Matrix Max-Log-MAP Algorithm 1

$$\Lambda(d_k) \approx 2\bar{\alpha}_0 \nabla \bar{\Gamma}(R_1) \nabla \bar{\Gamma}(R_2) \nabla \cdots \nabla \bar{\Gamma}(R_{k-1}) \nabla' \bar{\Gamma}_1(R_k) \nabla \bar{\Gamma}(R_{k+1}) \nabla \bar{\Gamma}(R_{k+2}) \nabla \cdots \nabla \bar{\Gamma}(R_N) \nabla \bar{\beta}_N{}^{\mathrm{T}} - \bar{\alpha}_N \nabla \bar{\beta}_N{}^{\mathrm{T}} \quad (16)$$

we can obtain another matrix Max-Log-MAP algorithm can be obtained as follows:

Matrix Max-Log-MAP Algorithm 2

1) Initialize $\bar{\alpha}_0$ and $\bar{\beta}_N$ as in step 1) of the Matrix Max-Log-MAP Algorithm 1.
2) For each observation R_k, $\bar{\Gamma}(R_k)$ and $\bar{\Gamma}_1(R_k)$ are computed using (4) and (5), respectively. Then compute $\bar{\alpha}_k (= \bar{\alpha}_0 \nabla \bar{\Gamma}(R_1) \nabla \bar{\Gamma}(R_2) \nabla \cdots \nabla \bar{\Gamma}(R_k))$, using (14), and at meantime, compute $\bar{\alpha}_0 \nabla \bar{\Gamma}_1(R_1) \nabla \bar{\Gamma}(R_2) \nabla \cdots \nabla \bar{\Gamma}(R_k)$, $\bar{\alpha}_0 \nabla \bar{\Gamma}(R_1) \nabla' \bar{\Gamma}_1(R_2) \nabla \bar{\Gamma}(R_3) \nabla \cdots \nabla \bar{\Gamma}(R_k)$, ... , $\bar{\alpha}_0 \nabla \bar{\Gamma}(R_1) \nabla \cdots \nabla \bar{\Gamma}(R_{k-1}) \nabla \bar{\Gamma}_1(R_k)$ in parallel.
3) After receiving the complete sequence R_1^N, compute $\bar{\alpha}_N \nabla' \bar{\beta}_N{}^{\mathrm{T}} (= \bar{\alpha}_0 \nabla \bar{\Gamma}(R_1) \nabla \bar{\Gamma}(R_2) \nabla \cdots \nabla \bar{\Gamma}(R_N) \nabla \bar{\beta}_N{}^{\mathrm{T}}$, and at the same time, compute $\bar{\alpha}_0 \nabla \bar{\Gamma}_1(R_1) \nabla \bar{\Gamma}(R_2) \nabla \cdots \nabla \bar{\Gamma}(R_N) \nabla \bar{\beta}_N{}^{\mathrm{T}}$, $\bar{\alpha}_0 \nabla \bar{\Gamma}(R_1) \nabla' \bar{\Gamma}_1(R_2) \quad \nabla \bar{\Gamma}(R_3) \nabla \cdots \nabla \bar{\Gamma}(R_N) \nabla' \bar{\beta}_N{}^{\mathrm{T}}$, ... , $\bar{\alpha}_0 \nabla \bar{\Gamma}(R_1) \nabla \cdots \nabla \bar{\Gamma}(R_{k-1}) \nabla' \bar{\Gamma}_1(R_N) \nabla' \bar{\beta}_N{}^{\mathrm{T}}$ in parallel. Then compute $\Lambda(d_k)$, $k = 1, 2, \ldots, N$, using (15).

Its implementation architecture is shown in Figure 4.

3 Algorithm Analysis and Comparison

Compared with the conventional Max-Log-MAP algorithm, the presented matrix Max-Log-MAP algorithms have following major features:

Computation Time

The greatest advantage of our matrix algorithms is that they can considerably speed up the decoding operations. This is because after matrixing the Max-Log-MAP decoder, we can do matrix operations between different rows and corresponding columns in parallel, and for the operation between any one row and corresponding column, we can do the operations of different element pairs in parallel. Therefore, the operations between any M–dimension vector and any $M \times M$ matrix can save time to $1/M^2$; the operations between any $M \times M$ matrixes can save time to $1/M^3$. Also, a lot of decoding procedures originally carried out successively in the conventional Max-Log-MAP algorithm can be accomplished in parallel, such as the steps 2) and 3) in both matrix algorithms.

Computational Complexity

Like the general logarithmic MAP like algorithms, the invented matrix Max-Log-MAP algorithms maintain much lower computational complexity than the original MAP algorithm. This is because these logarithmic MAP like algorithms transform the complex operations carried out in the original MAP into simple operations, such as additions and maximum functions. Most notably, as mentioned above, they avoid non-linear function computations for branch transition probabilities, all of which significantly reduce the VLSI implementation difficulty. Compared with the conventional Max-Log-MAP algorithm, the matrix Max-Log-MAP algorithms bear more complexity that mainly caused by the series operations between $\bar{\Gamma}(R_k)$ s (including $\bar{\Gamma}_i(R_k)$, $i = 0,1$). But it is these series operations that makes our matrix algorithms achieve

Note: The shift register with length of $N+1$ is initialized by (α_0 , #, ... , #), where $-\infty < \# < +\infty$.

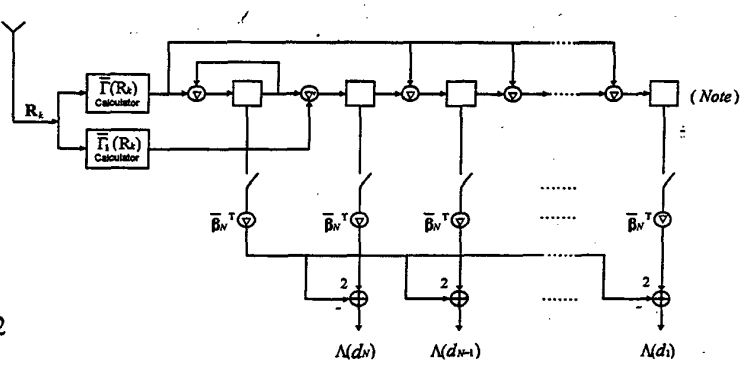


Figure 4 Implementation architecture of the Matrix Max-Log-MAP Algorithm 2

Table 1 Computational complexity of matrix Max-Log-MAP, Max-Log-MAP and MAP algorithms

Operation \ Algorithm	MAP*[1]	Max-Log-MAP	Matrix Max-Log-MAP 1	Matrix Max-Log-MAP 2
Addition	$4MN-2N$	$8MN$	$2M^2N^2-6M^2N+8MN$	MN^2+3MN
Subtraction	0	N	N	N
Maximum Function	0	$4MN-2N$	$3M^2N^2/2-9M^2N/2+4MN-2N$	$MN^2/2+3MN/2-N$
Multiplication	$6MN$	0	0	N^{*2}
Division	$2MN+N$	0	0	0
Non-linear Function	$4MN$	None	None	None

extremely higher decoding speed than the conventional Max-Log-MAP. The Max-Log-MAP Algorithm 2 is less complex than the Max-Log-MAP Algorithm 1 and faster than the latter since it totally skips the calculation procedures for the backward recursion functions and at the same time has a more simplified computational structure for $\Lambda(d_k)$s. The Table 1 lists a comparison on the computational complexity between the two matrix Max-Log-MAP algorithms, the conventional Max-Log-MAP and MAP algorithms.

Memory Capacity

The conventional MAP and Max-Log-MAP algorithms need a substantial amount of storage, which is usually MN. Using shift registers, we proposed simple implementation architectures that only need $2N$ and $N+1$ storage elements as shown in Figure 3 and Figure 4, respectively. The implementation schemes can also effectively reduce the memory capacity and simplify the data accesses and transfers.

Operation Mode

The most feature of our matrix algorithms is their facilitating the implementation of efficient VLSI circuits. Acting on $M\times M$ matrix (or M-dimension vector), our algorithms may be regarded as a VLSI oriented algorithm. The novel parallel operation constructions devised in the algorithms provide a feasible way to real-time implement the logarithmic MAP like decoding for turbo codes in special-purpose parallel processing VLSI hardware architectures.

Acknowledgement

This work has been supported, in part, by grants from the New Jersey Center for Wireless Telecommunications (NJCWT), the National Science Foundation (NSF) and Mitsubishi Electric Research Laboratories, Murray Hill, NJ (MERL).

References

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284-287, March 1974.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes (1)," in *Proc IEEE Int. Conf. on Commun,* Geneva, Switzerland, May 1993, pp. 1064-1070.

[3] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," 1995 IEEE Int. Conf. on Commun. (ICC'95), Seattle, 1995, vol. 2, pp. 1009 –1013.

[4] K. A. Wen, T. S. Wen, and J. F. Wang, "A new transform algorithm for Viterbi decoding," *IEEE Trans. Commun.*, vol.38, no. 6, pp. 764-772, June 1990.