J. Res. Inst. Indus. Tech.
Cheju Nat. Univ., 5 : 33-42, 1994.

濟州大 産技研報, 5 : 33-42, 1994.

콘볼루션 부호를 위한 비터비 복호 변환 알고리즘

전영희*, 김장형**, 이용학***

The Viterbi Decoding Transform Algorithm for Convolution Codes

Young Hee Jun*, Jang Hyung Kim**, and Yong Hak Lee***

ABSTRACT

In many applications researchers have been interested in implementation of Viterbi decoding algorithm for convolution codes. But it is problems that long constraint length codes need many storages and hardware implementation is complicated.

In this paper, several stages of trellis diagram are integrated and this procedures are formulated into simple matrix operations. Also backtracking decoding procedures is eliminated. This algorithm is called a new Viterbi transform (VT).

The number of the trellis search can be reduced using the VT. Because of local memory access, the storages can be taken in. Long constraint length codes can be decoded by combining the unit module processors.

I. 서 론

위성 통신이나 이동 통신 등에서 이용되고 있는 아날로그 시스템은 채널 용량의 한계성과 정보의 신뢰성이라는 관점에서 많은 문제가 있어 왔다. 이를 보완하는 디지탈 통신 시스템은 채널 용량의 한계를 극복하고 정보의 신뢰성을 보장하기 위하여 아날로그 신호를 이산화하는 소스코딩과 채널상의 에러를 정정하기 위한 채널 코딩의 과정을 거쳐 데이타를 전송한다.[1]

채널상에서 발생하는 에러 정정을 위한 채널 코딩은 메모리가 필요없는 블럭 코딩과 메모리를 필요로 하는 콘볼루션 코딩으로 크게 나눌 수 있다. 블럭 코딩은 패리티를 추가하는 방법으로 선형 블

* 대학원 통신공학과
** 정보공학과 부교수
*** 통신공학과 교수

럭 부호, 순환부호, BCH 부호 등이 제안되었으며, 콘볼루션 코딩은 입력 데이타를 메모리에 있는 기존의 데이타와 상관 관계를 갖게 부호화 하는 방법이다.[2][3] 블럭 코딩에 비하여 에러 정정 효율이 우수한 콘볼루션 부호가 주로 사용되고 있다. 콘볼루션 부호는 복호시에 많은 양의 메모리를 필요로 하며, 복호화 지연이 발생하기 때문에 실시간 처리가 가능한 복호 알고리즘이 필요하게 되었다. 이에 따라 콘볼루션 부호의 최우 복호 알고리즘인 비터비 알고리즘(VA)이 제안되었는 데, 이 알고리즘은 위성 통신, 문자와 음성인식 등의 여러 분야에서 응용되고 있으며, 그 응용범위가 증가되고 있는 추세이다.[4]-[7] 그러나 VA의 문제점은 구속장의 길이가 긴 부호에서 지나치게 많은 기억장소를 필요로 하고 하드웨어의 구현이 복잡하다는 것이다. 하드웨어의 구현을 효율적으로 하고 계산시간을 줄이기 위하여 VA에 관한 수많은 알고리즘들이 발표되었다.[8]-[13] 본 논문은 콘볼루션 부호와 비터비 알고리즘을 분석하였고, 격자도의 여러 단계를 통합하는 원리를 응용하여 비터비 알고리즘을 변환한 비터비 변환(Viterbi Transform; VT) 알고리즘을 제안했다.

본 논문에서는 비터비 알고리즘을 대수학적으로 보완하여, 여러 단계의 복호과정들을 간단한 행렬 연산으로 변환하였다. 새로운 알고리즘은 전체 메모리를 접근하는 것이 아니라 부분 메모리를 접근하도록 제한하고 역추적 과정을 제거하기 때문에, 데이타 전송과 억세스 야기되는 VLSI 회로의 시간 소비를 감소시킬 수 있다.

Ⅱ. 비터비 복호를 위한 비터비 알고리즘의 변환

Ⅱ-1. 콘볼루션 부호와 비터비 알고리즘

1. 콘볼류션 부호

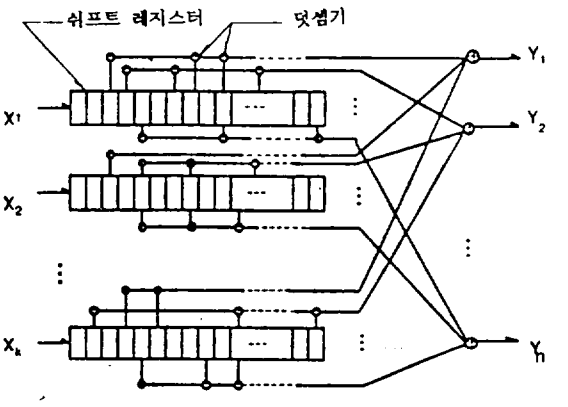

Fig 1 Convolutional encoder.

Fig 1에 보이는 바와 같이 (n, k, m)인 콘볼루션부호기는 k개의 입력, X_1, X_2, …, X_k으로 하여 n개의 출력, Y_1, Y_2, …, Y_n을 보내는 유한 상태기다. 각각의 입력노드와 연결된 부호기는 쉬프트레지스터와 Modulo-2 덧셈기로 구성되어 있다.[2] 입력 비트수인 구속장 K와 입출력 비트수의 비인 부호비 R에 의하여 부호기의 출력과 구조가 결정된다. 덧셈기의 연결점은 생성벡터나 생성 다항식으로 표현하는 대수학적 방법과 상태도나 격자도 등으로 나타내는 위상학적 방법이 있다.[1][2] 상태도를 이용한 콘볼루션 부호기의 기술은 콘볼루션 부호기의 레지스터 값을 하나의 상태로 하여 각입력 값에 따른 상태의 변화를 나타낸 것이며, 상태의 변화를 이산적인 시간축의 함수로 나타낸 것이 격자도이다.

2. 비터비 알고리즘

(n, k, m) 부호의 격자도에는 $N=2^{km}$개의 상태가 있다. 상태 S_i에서 다음 단계의 상태 S_j를 연결하는 가지를 $e_{i,j}$로 나타낸다. (Fig 2)

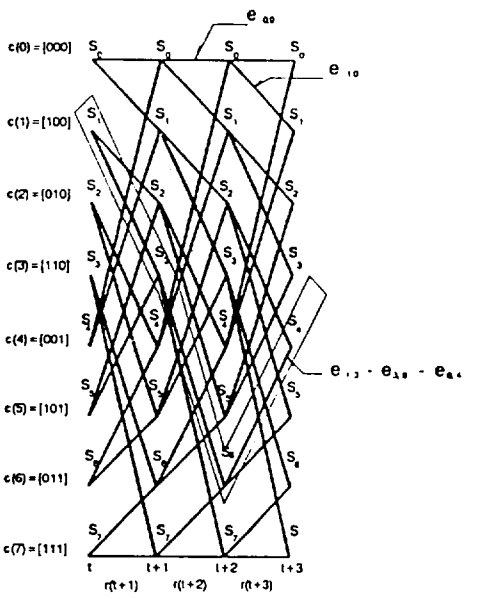

Fig 2 Trellis diagram of an eight state Convolution code.

비터비 알고리즘은 생존 metric을 계산해서 격자도의 최단경로를 동적으로 찾아간다. 시점 t의 S_j에 연결되는 후보경로는 2^k개가 있고, 그 경로들 중에서 시점 t-1의 상태 S_i와 연결되는 생존경로, 그 S_i와 시점 t의 S_j간의 연결가지 $e_{i,j}$로 구성되어 있다.

각 시점에서 필요한 연산은 생존 metric을 찾아서 S_j에 대한 모든 후보경로들 중에서 생존경로를 결정하는 것이다. J(j)는 격자도상의 각 상태의 가지 $e_{i,j}$에 대하여 S_j와 연결되는 이전 단계의 상태 S_i의 첨자들의 집합으로서 그 수는 2^k개가 된다. 예를 들면, 그림 2에서 시점 t+3의 상태 S_4의 J(4) = {2, 6}이다. S_j의 모든 후보경로에 대한 경로 metric pm(i, j, t)는 유사측정 함수 f를 거쳐서 계산하고(i∈J(j)), 그것들 중에서 최단 경로 metric을 생존 metric sm(j, t)로서 선택한다. r(t)를 t번째 시점에서 수신된 n-비트부호어이고, w(i, j)는 임의의 상태로 전이될 때의 부호기의 출력으로서 $e_{i,j}$에 대응하는 부호어이다. pm(i, j, t)와 sm(i, t)는 다음과 같은 관계가 성립한다.

$$pm(i, j, t) = f\{r(t), w(i, j), sm(i, t-1)\} \quad (1)$$

$$sm(j, t) = X(pm(i, j, t))\ (= \max(pm(i, j, t);\ \forall i \in J(j))) \quad (2)$$

X는 최단경로 비교기이고, i∈J(j)인 모든 상태 S_i에 적용된다.

식(2)에서, S_j의 생존 metric을 계산할 때 관계있는 연산자는 r(t), {w(i, j)∖i∈J(j)}, {sm(i, t-1)∖i∈J(j)}이다.

간단하게 요약하면, 비터비 알고리즘은 복호기의 입력과 부호기에 의하여 주어지는 격자도의 기준값사이의 해밍거리인 가지 metric를 누적하여 경로 metric을 계산한다. 각 상태에서의 경로 metric은 들어오는 두개의 가지 중에서 전 단계의 경로 metric과 생존 metric을 합한 것 중 작은 것을 선택하여 얻어진다. 선택된 가지를 생존자라고 하며 가장 짧은 경로 metric을 갖는 상태에서 시작하여 생존자를 역추적함으로서 복호기의 출력을 결정한다.

Ⅱ-2. 격자도의 통합

위상학적으로 수정보완하여, 격자도의 p개의 단계를 하나로 통합하여 1단 격자도로 변형할 수 있다. (n, 1, 3) 부호의 격자도에 대한 변형을 Fig 3(a)와 (b)에서 설명한다. Fig 3(a)는 두 단계씩 통합한 것이고 각 가지는 원래 격자도 (Fig 2)의 2-가지경로와 일치한다. Fig 3(b)는 3단계씩 통합하여 3-가지경로를 나타내고 있다. Fig 2의 테두리 속에 있는 경로들을 참조해 보면, pm(1, 3, t+1)을 (t+1)시점에서 S_3에 대한 생존

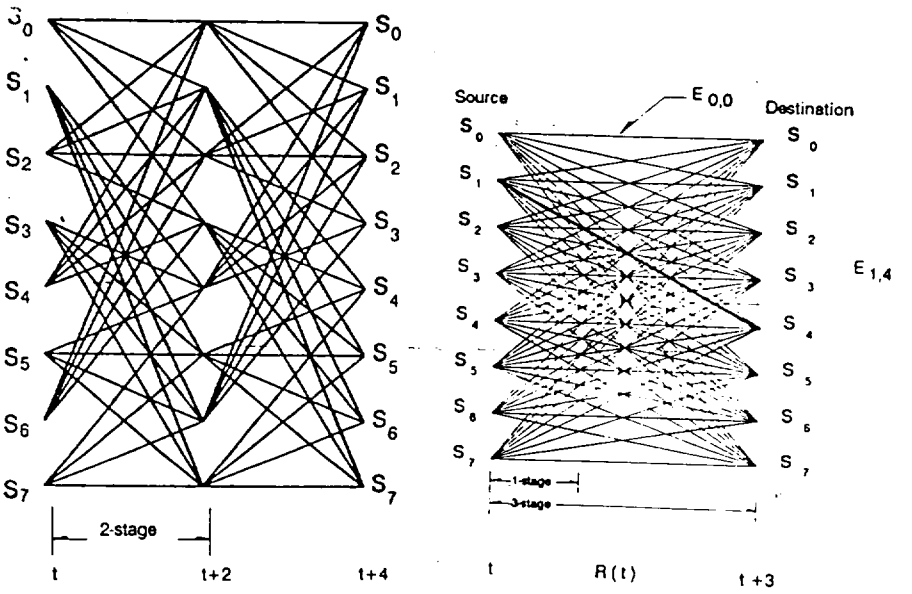

Fig 3. (a) The integrated trellis diagram (p=2) (b) The integrated trellis diagram (p=3)

metric으로 결정하고 pm(3, 6, t+2)를 (t+2) 시점에서 S_6에 대한 생존 metric으로 결정한다. 그리고 $e_{1,3}$, $e_{3,6}$, $e_{6,4}$로 구성된 3가지 경로, $p_{6,4,t+3}$는 (t+3) 시점의 S_4의 후보경로중 하나이다.

일반적인 VA에서 pm(6, 4, t+3)은 다음 식처럼 단계마다 반복해서 계산된다.

pm(6, 4, t+3)
= f(r(t+3), w(6, 4), sm(6, t+2))
= f(r(t+3), w(6, 4), X(pm(i, 6, t+2)))
= f(r(t+3), w(6, 4), pm(3, 6, t+2))
= f(r(t+3), w(6, 4), f(r(t+2), w(3, 6), sm(3, t+1)))
= f(r(t+3), w(6, 4), f(r(t+2), w(3, 6), X(pm(i, 3, t+1)))..).

각 단계마다 동적처리를 하여, $i \in J(j)$를 만족하는 모든 S_i중에서 생존 metric은 S_j의 생존 metric을 갱신하는데 이용한다. 이것은 VLSI회로에서 데이타를 억세스할 때 가장 많은 시간을 소비하도록 하는 주요인이다.

p개의 단계들을 하나의 단계로 통합하면, 격자도에서 m개의 연속하는 가지들을 한 개의 통합된 가지로 변형되고, 한번의 계산과정으로 m개의 연속하는 가지들의 metric합을 계산할 수 있다. 새로운 경로 metric을 형성하기 전에 m개의 단계들에서 계산된 생존 metric과 합쳐질 수 있다면, 그때 각 생태에 대한 생존 metric은 각 p단계마다 갱신될 것이고 데이타 엑세스의 수는 p번 감소될 수 있다.

그러나, 통합된 격자도에서 정의된 가지/경로/생존자 계산들이 원래 VA격자도에서 계산된 것들과 호환성이 있도록 하기 위해서, 통합된 격자도상의 가지는 원래 격자도의 p-가지 경로에 1대 1 대응해야 한다는 제한이 있다.

Ⅱ-3. 비터비 알고리즘 변환

VA의 동적인 복호과정을 $E_{i,j}$로 수정하여 성분 가지에 대한 데이타 억세스와 데이타 전송을 제거한다. 이를 증명하기 위하여 (n, l, m) 콘볼루션 부호에 대한 VT를 기술하고 부호처리를 논한다.

1. 상태 부호 정의

(n, l, m) 인 부호는 하나의 입력, m개의 레지스터, n개의 출력이 있다. 레지스터의 수가 m이기 때문에, 부호기의 조합 가능한 상태수는 $N=2^m$이다. 각 상태 S_j를 j의 이진표현인 m-비트 상태 부호로 나타낸다. 즉,

$C(j)=S_j$의 상태부호=j의 이진표현

2. 가지부호생성

1) 1단 가지 부호 생성 공식

부호기의 주요 기능은 S_i에서 S_j로 가능한 상태 전이들에 따라서 가지부호 w(i, j)를 생성하는 것이다. 부호과정을 근거로 하여, 가지 부호를 생성하기 위하여 오퍼랜드벡터, in(i, j)는 m개의 레지스터 내용(C(i))과 S_i에서 S_j로 상태 전이를 야기시키는 입력비트의 내용으로 구성되어 있다. 입력비트가 부호기의 왼쪽부터 적재되므로 m개의 레지스터의 비트 조합은 C(i)에서 C(j)로 변환된다. C(j)의 가장 왼쪽 비트, LMB(j)는 S_i에서 S_j로 상태 전이를 일으키는 입력비트이다. 즉,

$$in(i,j)=[LMB(j)\backslash C(i)] \quad (4)$$

여기서 [\]는 연속을 의미하고 w(i, j)는 식(5)와 같이 일반화된다.

$$w(i,j)=in(i,j)*G \quad (5)$$

여기서 *는 GF(2)에서 정의한 행렬곱이다. G는 부호를 정의하는 $k(m+1)\times n$행렬이다.

$$G=\begin{bmatrix} G_0 \\ G_i \\ \cdots \\ G_m \end{bmatrix}$$

$$G_r=\begin{bmatrix} g^1_{1,r} & g^2_{1,r} & g^3_{1,r} & \cdots & g^n_{1,r} \\ g^1_{2,r} & g^2_{2,r} & g^3_{2,r} & \cdots & g^n_{2,r} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g^1_{k,r} & g^2_{k,r} & g^3_{k,r} & \cdots & g^n_{k,r} \end{bmatrix}$$

요소 $g^j_{i,r}$는 g^j_i의 r번째 비트, X_i와 Y_j에 관한 생성 기열이다.[2] 예를 들어 (3, 1, 3) 콘볼루션부호를 다음과 같이 정의하면,

$$G=\begin{bmatrix} 110 \\ 011 \\ 101 \\ 001 \end{bmatrix}$$

$$w(1,3)=[LMB(3)\backslash C(1)]*G=[1\backslash 100]*G=[101] \quad (6)$$

$$w(3,6)=[LMB(6)\backslash C(3)]*G=[0\backslash 110]*G=[110] \quad (7)$$

$$w(6,4)=[LMB(4)\backslash C(6)]*G=[0\backslash 011]*G=[100] \quad (8)$$

2) m단 가지 부호 생성 공식

$E_{i,j}$에 대하여, 조합된 m개의 단계들과 m개의 레지스터의 내용이 C(j)로 변하는 동안 m개의 입력비트들이 입력된다. 바꾸어 말하면, C(j)는 m개의 입력 비트들로 이루어져 있다. $E_{i,j}$에 대한 m단 가지 부호 W(i, j)를 계산할 때 다음 처럼 C(j)를 연산자로 이용한다.

$$IN(i,j)=[C(j)\backslash C(i)] \quad (9)$$

$$W(i,j)=IN(i,j)*Gm \quad (10)$$

Gm은 다음처럼 앞의 열보다 한 행 밑으로 나열되고 있는 확장된 $2m\times mn$ 행렬이다.

$$Gm=\begin{bmatrix} G_0 & & \\ G_1 & G_0 & \\ G_2 & G_1 & \\ \vdots & G_2 & \\ \vdots & \vdots & \\ G_{m-1} & \vdots & G_0 \\ G_m & G_{m-1} & G_1 \\ & G_m & G_2 \\ & & \vdots \\ & & \vdots \\ & & G_{m-1} \\ & & G_m \end{bmatrix}$$

식(10)에서 구해진 확장된 m단 부호는 m개의 요소 가지들의 모든 가지 부호들로 구성되어 있다. 앞의 예에 대한 확장된 생성행렬은 다음과 같다.

$$Gm=\begin{bmatrix} 110 & & \\ 011 & 110 & \\ 101 & 011 & 110 \\ 001 & 101 & 011 \\ & 001 & 101 \\ & & 001 \end{bmatrix}$$

$W(1,4)=[001\backslash 100]*Gm=[100110101]$임을 알 수 있다. 식(6), (7), (8)과 비교해 보면, $E_{1,4}$는 $e_{1,3}, e_{3,6}, e_{6,4}$로 구성되어 있고, $W(i,j)$는 $w(1,3)$, $w(3,6)$, $w(6,4)$들이 역으로 연결되어 있다. 즉,

$$W(1,4)=[w(6,4)\backslash w(3,6)\backslash w(1,3)]$$

3. m단 가지 부호 생성을 위한 수치적 보완

좀 더 축소하기 위하여 Gm을 각각 m x nm행렬인 Gm_u와 Gm_L 분리한다. Gm_u와 Gm_L은 각각 Gm의 상반과 하반이다. 상태 S_i에 따라서 달라진다. 상태 S_j로 들어오는 안쪽 가지에 대한 $W(i,j)$를 계산할 때 $C(j)$와 Gm_u의 부분 행렬 곱은 고정되고, 식(10)에서 $T(j)$로 분리할 수 있다. 즉,

$$W(i,j)=IN(i,j)*Gm$$

$$=[C(j)\backslash C(i)]*\left(\frac{Gm_u}{Gm_L}\right)$$

$$=[C(j)*Gm_u]\oplus[C(j)*Gm_L] \quad (11)$$

$$=[T(j)\oplus[C(i)*Gm_L]$$

⊕는 배타적 OR연산자이다. 증가된 차원을 피승수벡터에 더해서 식(11)을 식(12)와 같이 간단하게 줄일 수 있다.

$$W(i,j)=[C(i)\backslash 1]*\left(\frac{Gm_L}{T(j)}\right) \quad (12)$$

4. m단 가지 Metric계산

이진 대칭 채널(BSC)의 경우, $w(i,j)$와 $r(t)$는 n차 벡터이다. 가지 metric $bm(i,j,t)$를 $w(i,j)$와 $r(t)$간의 해밍거리로부터 구할 수 있다. $bm(i,j,t)$가 $r(t)$와 t시점의 모든 부호어 간의 해밍거리들 중에서 최소 해밍거리라면 최우 복호기는 $w(i,j)$를 전송 부호어로 선택한다. 가지 metric을 식(13)처럼 내적연산 "x"로 표현할 수 있다.

$$bm(i,j,t)=[in(i,j)\backslash 1]*\left(\frac{G}{r(t)}\right)\times I^T \quad (13)$$

1는 n차 단위 벡터 $I=[111\cdots 1]$이다. 식(13)을 $E_{i,j}$에 대하여 확장하면, 연속하는 m단에 수신된 부호를 고려해야 한다. $R(t)=[r(t)\backslash r(t-1)\backslash\cdots r(t-m+1)]$라고 하면, $E_{i,j}$의 m단 가지 metric은 다음과 같다.

$$Bm(i,j,t)=(W(i,j)\oplus R(t))\times I_m^T$$

$$=([C(i)\backslash 1]*\left(\frac{Gm_L}{T(j)}\right)\oplus R(t)\times I_m^T \quad (14)$$

$$=([C(i)\backslash 11]*\left(\frac{Gm_L}{\frac{T(j)}{R(t)}}\right)\times I_m^T$$

I_m은 I로 부터 확장된 nm차원 단위벡터이다.

5. m단 경로 metric계산

가지 metric을 계산한 후, 경로 metric을 계산해야 한다. 경로 metric, pm(i, j, t)은 bm(i, j, t)와 sm(i, t-1)의 합으로 다음처럼 정의한다.

$$sm(j, t+1) = X(pm(i, j, t+1))$$
$$pm(i, j, t+1) = f(r(t),\ w(i, j),\ sm(i, t)) = bm(i, j, t+1) + sm(i, t)$$

m단으로 통합되었을 때, 각 상태에 대한 J(j)는 0에서 N-1까지의 모든수를 포함한다. 통합된 격자도의 경로 metric와 생존 metric은 각각 Pm(i, j, t)와 Sm(j, t)로 나타낸다. Pm(i, j, t)를 계산하고 나서 연속으로 Sm(j, t)을 구하는데 i는 0에서 N-1까지의 X연산을 한다.

$$Sm(j, t) = X(Pm(i, j, t)) \tag{15}$$

$$Pm(i, j, t) = Bm(i, j, t) + Sm(i, t-m)$$
$$= [C(i)\backslash 11] * \begin{pmatrix} Gm_L \\ T(j) \\ Re(t) \end{pmatrix} \times I_m^T + Sm(i, t-m)$$
$$= [C(i)\backslash 11] * \begin{pmatrix} Gm_L \mid 0 \\ T(j) \mid 0 \\ Re(t) \mid 1 \end{pmatrix} \times [I_m \mid Sm(i, t-m)]^T \tag{16}$$

행렬 연산 $A[B|C] = A*B\times C^T$라고 정의하면,

$$Pm(i, j, t) = [C(i) \mid 11] \begin{pmatrix} Gm_L \mid \\ T(j) \mid \\ \hline R(t) \mid Sm(i, t-m) \end{pmatrix} \tag{17}$$

그러므로, 부호어 생성, 통합된 격자도의 각 상태에 대한 가지 metric과 경로 metric을 식(17)에 주어진 간단한 행렬 연산에 의해 구해진다.

6. 역추적(Backtracking) 제거

일반적인 비터비복호에서는 역추적 과정을 거쳐 최종 복호된 부호를 찾아내야 한다. 변환 알고리즘(VT)에서는 Sm(j, t)를 계산할 때, 최단 경로 metric의 전 상태에서 현 상태로 전이되도록 작용한 입력 비트들을 스택에 쌓아둔다. Sm(j, t)가 모든 상태에 대한 생존 metric들 중에서 최대값이라면, 쌓아 두었던 스택의 값을 직접 읽어 낼 수 있다. 즉 일반적인 VA에서 처리했던 역추적 과정은 제거된다.

Ⅲ. 시뮬레이션 결과

다음의 생성행렬을 가지고 (3, 1, 2) 부호를 예로 하여 VA알고리즘을 수치적으로 계산하고 VT 알고리즘을 시뮬레이션했다.

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

수신부호열 r(t) = [011 110 001 100 110 000 101 001 011 101]가 복호기에서 얻어진다고 가정하면, w(i, j)는 다음 Table 1과 같이 구해진다.

Table 1. Branch code

i, j	branch code	i, j	branch code
0, 0	0 0 0	1, 2	0 1 1
2, 0	1 0 1	3, 2	1 1 0
0, 1	1 1 0	1, 3	1 0 1
2, 1	0 1 1	3, 3	0 0 0

각 시점마다 가지, 경로, 생존 metric은 Table 2와 같다.

VT 알고리즘에서,

$$Gm = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$T(0) = [0\ 0] * \begin{pmatrix} 1\ 1\ 0\ 0\ 0\ 0 \\ 0\ 1\ 1\ 1\ 1\ 0 \end{pmatrix} = [0\ 0\ 0\ 0\ 0\ 0],$$

$T(1) = [1\ 1\ 0\ 0\ 0\ 0]$,

$T(2) = [0\ 1\ 1\ 1\ 1\ 0]$,

$T(3) = [1\ 0\ 1\ 1\ 1\ 0]$,

i는 0에서 N-1 범위에 대하여 Sm(i, 0)을 0으로 초기화하고 경로 metric은 다음과 같이 계산된다. VT의 경로, 생존 metric 계산들은 Table 3에 나타나 있다.

Table 2. Computation Result of VA

t		1	2	3	4	5	6	7	8	9	10
bm	0, 0	2	2	1	1	2	0	2	1	2	2
	2, 0			1	1	2	2	0	1	2	0
	0, 1	2	0	3	1	0	2	2	3		
	2, 1			1	3	2	2	2	1		
	1, 2		2	1	3	2	2	2	1	0	
	3, 2			3	1	0	2	2	3	2	
	1, 3		2	1	1	2	2	0	1		
	3, 3			1	1	2	0	2	1		
μm	0, 0	2	4	5	6	6	6	8	7	9	11
	2, 0			5	4	6	6	6	9	11	9
	0, 1	2	2	7	6	4	8	8	9		
	2, 1			5	6	6	6	8	9		
	1, 2		4	3	8	8	6	8	9	9	
	3, 2			7	4	4	8	8	9	9	
	1, 3		4	3	6	8	6	6	9		
	3, 3			5	4	6	6	8	7		
sm	0	2	4	5	4	6	6	6	7	9	9
	1	2	2	5	6	4	6	8	9	9	
	2		4	3	4	4	6	8	9		
	3		4	3	4	6	6	6	7		

Table 3. Computation Result of VT

t		1	2	3	4	5	6	7	8	9	10
Pm	0, 0		4		6		6		9		11
	1, 0				4		10		9		
	2, 0				6		6		7		9
	3, 0				8		6		9		
	0, 1		2		6		8		11		
	1, 1				6		10		9		
	2, 1				6		8		9		
	3, 1				10		6		9		
	0, 2		4		10		6		9		
	1, 2				4		10		9		
	2, 2				8		8		9		
	3, 2				6		8		11		
	0, 3		4		8		6		9		
	1, 3				4		8		7		
	2, 3				6		8		9		
	3, 3				6		6		9		
Sm	0		4		4		6		7		9
	1		2		6		6		9		
	2		4		4		6		9		
	3		4		4		6		7		

Ⅳ. 결 론

비터비 알고리즘(VA)에서는 구속장의 길이가 긴 부호의 경우 많은 기억장소를 필요로 하며, 그에 따른 하드웨어의 구현이 복잡하다는 단점이 있다. 이를 보완하기 위하여 본 논문에서는 VA의 격자도를 여러 단계 통합하는 비터비 알고리즘을 변환한 통합 알고리즘(VT)을 제안하였다.

위상학적으로 표현되어 왔던 VA를 수치적으로 공식화하여 여러 단계의 복호 과정들을 간단한 행렬연산으로 나타내었고, VT에서는 몇 개의 가지들을 통합한 후 행렬연산으로 공식화하였다. VA와 VT를 시뮬레이션한 결과 복호기의 출력이 동일함을 알 수 있었다. 그리고 계산 횟수는 2단계를 통합하였을 경우 VT가 VA보다 1/2배 만큼 줄었음을 확인할 수 있었다. VA에서는 역추적 과정을 거쳐서 복호기의 출력을 결정하므로 데이타 전송이나 엑세스시에 많은 시간이 필요한 반면, VT에서는 부분 메모리를 접근하도록 제한되어 있고 역추적 과정이 제거되기 때문에 데이타 전송과 억세스할 때 야기되는 VLSI회로의 시간 소비를 감소시킬 수 있다.

참 고 문 헌

[1] G. Clark, J. Cain, Error-Correction Coding for Digital Communication, Plenum Press : New-York, 1981.

[2] Man young Rhee, Error Correction Coding Theory, McGraw-Hill, 1989.

[3] S. Lin and D.J. Costello, Jr., Error Control Coding : Fundamentals and Applications, Englewood Cliffs, NJ, Prentice Hall, 1983 ch 11.

[4] G. D. Forney, Jr., "Convolutional codes Ⅱ : Maximum likelihood decoding", Inf. Contr., vol.25, pp.222-266, July 1974.

[5] S. Haykin and J.P. Reilly, "Maximum likelihood receiver for low angle tracking radar, Part 1 : The symmetric case", IEE Proc., vol.129, Pt.F, pp.261-272, 1982.

[6] J.P. Reilly and S. Haykin, "Maximum-likelihood receiver for low angle tracking radar, Part 2 : The nonsymmetric case", IEE Proc. vol.129, Pt.F, pp.261-272, 1982.

[7] J.K. Skwirzynski, The Impact of Processing Techniques on Comminications. The Netherlands : Martinus Nijhoff, 1985, pp.159-192.

[8] J.A. Heller and I.M. Jacobs, "viterbi decoding for satellite and space communication", IEEE Trans. Commun. Technol., vol. COM-19, pp.835-848, Oct. 1971.

[9] P.G. Gulak and E. Shwedyk, "VLSI structures for Viterbi receivers : Part 1-General theory and applications", IEEE J. Select. Areas Commun., vol. SAC-4, Jan. 1986.

[10] S. Mohan and A.K. Sood, "A multiprocessor architecture for the (M, L)-algorithm suitabie for VLSI implementation", IEEE Trans. Commun., vol. COM-34, Dec. 1986.

[11] T. Ishitani, K. Tansho, N. Miyahara, S. Kubota, and S. Kato, "A scarcestate transition Viterbi decoder VLSI for bit error correction", IEEE J.

Solid-State Circuits, Vol. SC-22, Aug. 1987.

[12] C.Y. Chang and K. Yao, "Viterbi decoding by systolic array", in 23rd Ann. Allerton Conf. Commun., Contr., Comput., 1985, pp. 430-439.

[13] R.M. Fano, "A heuristic discussion of probabilistic decoding", IEEE Trans. Inform. Theory, vol. IT-9, pp. 64-73, Apr. 1963.

[14] F. Jelinek, "A fast sequential decoding algorithm using a stack", IBM J. Res. Deelop., vol. 23, pp. 675-685, Nov. 1969.