MAP(Maximum A Posteriori)복호 알고리즘을 이용한 MAP Decoder의 설계

정 득 수*, 송 오 영*
*중앙대학교 전자전기공학부
e-mail : song@jupiter.cie.cau.ac.kr

Design of A MAP Decoder with MAP(Maximum A Posteriori) Algorithm

Deuk Soo Jung*, Ohyoung Song*
*School of Electrical & Electronics Engineering, Chung-Ang University

요약

본 논문은 MAP(Maximum A Posteriori) 복호 알고리즘을 이용한 MAP Decoder의 설계에 관해 다룬다. 채널코딩기법은 채널을 통해서 디지털 정보를 전송할때 신뢰성을 제공하기 위해서 사용되어 진다. 즉 수신단에서 수신된 정보의 오류를 검사하고 수정하기 위한 목적으로 송신단에서는 디지털 정보에 부가 정보를 첨가해서 전송하게 된다. 그래서 무선 이동 통신에서 성능이 우수한 채널코딩기법은 우수한 통신 품질을 위해서는 필수적이라고 할 수 있다. 최근에 Shannon의 한계에 매우 근접한 성능으로 많이 알려진 오류정정부호로 터보코드가 발표되었고 많은 연구가 진행되고 있다. 터보코드의 부호기로는 RSC(recursive systematic convolutional)코드가 사용되며 디코딩 알고리즘으로는 주로 MAP 복호 알고리즘을 사용한다. 본 논문에서 제안된 MAP 복호기는 하드웨어로 구현하기 위해서 변형된 LOG-MAP 복호 알고리즘을 이용하였고 터보디코더의 반복 복호에 이용할 수 있다.

Ⅰ. 서론

본 논문은 MAP(Maximum A Posteriori) 복호 알고리즘을 이용한 MAP Decoder의 설계에 관해 다룬다. 본문의 구성은 2장에서는 터보코드가 회귀되는 정보를 이용한 반복 복호의 이득을 얻기 위해서 Bahl et. al. 알고리즘[7]을 변형한 MAP(Maximum A Posteriori) 복호 알고리즘[8]에 대하여 설명한다. 3장에서는 2장에서 설명한 MAP 복호 알고리즘을 하드웨어로 구현하기 위한 LOG-MAP 복호 알고리즘을 이용하여 MAP 복호기의 구조를 제시하고 동작원리에 관해서 설명한다. 마지막으로 4장에서는 결과를 고찰하고 결론을 맺는다.

2. 본론

2.1 MAP 복호기 설계

MAP 복호기는 크게 가지 메트릭을 계산하는 BMC(branch metric calculator)블록, 역방향 메트릭을 계산하는 RSMC(reverse state metric calculator)블록, 순방향 메트릭을 계산하는 FSMC(forward state metric clculator)블록 그리고 LLR값을 계산하는 블록인 LLRC(log likelihood ratio calculator)로 구성된다. 여기에 E 함수 테이블 블록이 추가되며 RSMC가 역방향으로 계산되기에 BMC의 결과를 저장하는 공간인 BMS(BM store) 블록과 RSMC의 결과를 저장하는 RSMS(RSM store) 블록이 필요하다. E 함수 테이블 블록은 가지 메트릭을 계산하는 과정을 제외하고는 모든 계산 블록에서 참조가 되기 때문에 적절한 제어

신호가 필요하게 된다. MAP 복호기에서는 메모리의 양을 줄이기 위해서 역방향 메트릭값만 저장하게 되고 순방향 메트릭은 따로 저장할 필요 없이 역방향 메트릭값을 참조해서 LLR을 계산하도록 구현되었다. 또한 MAP 복호기의 구현을 간단하게 하기 위해서 LOG-MAP알고리즘에 사용되는 E 함수를 새롭게 정의하였으며 이에 따른 메트릭값을 새롭게 유도하였다. 그림 3.1는 전체 MAP복호기의 블록을 표시한 그림이다.

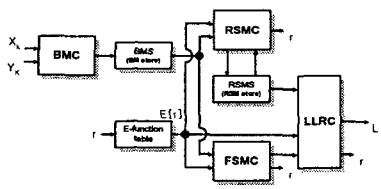

그림 3.1 MAP 복호기 구조

2.1.1 Log-MAP 알고리즘을 위한 E 함수의 정의 및 Lookup Table

본 논문에서는 MAP 복호 알고리즘을 하드웨어로 구현할 수 있도록 단순화시키기 위해서 E 함수를 식 (1)과 같이 정의하였다.

$$x\ E\ y=-\frac{1}{L_c}\ln(e^{-L_cx}+e^{-L_cy}),\quad (1)$$

$$L_c\doteq\frac{4E_c}{N_o},\quad (2)$$

$$\sigma^2=\frac{N_o}{2E_c},\ \frac{1}{\sigma^2}=\frac{E_c}{N_o/2},\quad (3)$$

$$\begin{aligned}x\ E\ y &= -\frac{1}{L_c}\ln(e^{-L_cx}+e^{-L_cy})\\ &= -\frac{1}{L_c}\times\ln(e^{-L_cx}(1+e^{-L_cy+L_cx}))\\ &= x-\frac{1}{L_c}\ln(1+e^{L_c(x-y)})\\ &= y-\frac{1}{L_c}\ln(1+e^{L_c(y-x)})\\ &= \min(x,y)-\frac{1}{L_c}\ln(1+e^{-L_c|y-x|}),\end{aligned}$$

(4)

2.1.2 BMC(branch metric calculator) 블록

3.1에서 정의된 E 함수를 이용하기 위해서 가지 메트릭 $\delta_i(R_k,m)$에 $-\frac{1}{L_c}$을 곱하고 로그를 취하면 식 (5)과 같이 표현된다.

$$\begin{aligned}D_i(R_k,m) &= -\frac{1}{L_c}\ln\delta_i(R_i,m)\\ &= -(x_k i+y_k Y_k^i(m)),\end{aligned}\quad (5)$$

여기서 i는 입력 비트를 나타내고 $Y_k^i(m)$는 입력 비트에 대한 패러티 비트를 나타낸다. 입력 i가 0과 1의 값을 가지고 $Y_k^i(m)$도 0과 1의 값이므로 $D_i(R_k,m)$의 값은 각 x_k, y_k에 대해서 4가지 경우의 값을 가지게 된다. BMC 블록은 아래 그림과 같이 x_k, y_k을 입력으로 받아서 delta0, delta1, delta2, delta3의 4가지 $D_i(R_k,m)$값을 생성해서 BMS(BM store) 블록에 값을 보내고 메모리에 저장되게 된다. 그러므로 $4N$ (N은 메시지 개수)개의 메모리가 필요하게 된다. 메모리에 저장된 값은 입력 비트와 패러티 비트를 이용해서 참조할 수 있게된다. 식 (5)을 이용하여 BMC블록을 구성하면 그림 3.2와 같다.

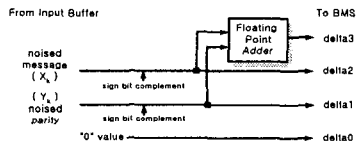

그림 3.2 BMC 블록

2.1.3 RSMC(reverse state metric calculator) 블록

가지 메트릭과 같은 식으로 역방향 메트릭 $\beta_k^i(m)$에 $-\frac{1}{L_c}$을 곱하고 로그를 취하면 식 (6)과 같이 표현된다.

$$\begin{aligned}B_k^i(m) &= -\frac{1}{L_c}\ln\beta_k^i(m)\\ &= E_{j=0}^{1}\ B_{k+1}^j(S_f^i(m))+D_j(R_{k+1},S_f^i(m)),\end{aligned}$$

(6)

우선 i = 0, 1에 대해 $\beta_{N-1}^i(S_f^i(0))=1$ 이고, 그 외 다른 모든 m과 i에 대해 $\beta_{N-1}^i(m)=0$으로 초기화를 시키므로 로그를 취하면 $B_{N-1}^i(S_f^i(0))$는 0이 되고 그 외 다른 모든 m과 i에 대해서는 무한대 값으로 초기화가 된다. 역방향 메트릭의 경우에는 $S_f^1(m)=S_f^0(m')$일 때 $\beta_k^1(S_f^1(m))=\beta_k^0(S_f^0(m'))$ 이므로 i가 0인 경우

만 계산을 하고 $S^1_f(m)$는 $S^0_b(S^1_f(S^0_f(m')))$인 관계를 이용하여 구할 수 있다. 이렇게 하면 배열의 크기를 절반으로 줄일 수 있다. 식 (6)에서 i는 0일 때만 계산하는 식으로 나타내면 아래와 같이 식 (7)으로 전개할 수 있다.

$$B^0_k(m) = B^0_{k+1}(S^0_f(m)) + D_0(R_{k+1}, S^0_f(m)) E$$
$$B^0_{k+1}(S^0_b(S^1_f(S^0_f(m)))) + D_1(R_{k+1}, S^0_f(m)), \quad (7)$$

식 (7)을 이용하여 역방향 메트릭을 구하기 위한 RSMC블록을 나타내면 그림 3.3과 같이 구성할 수 있다.

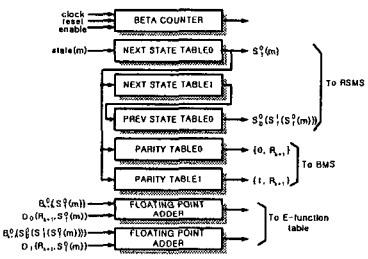

그림 3.3 RSMC 블록

2.1.4 FSMC(forward state metric calculator) 블록

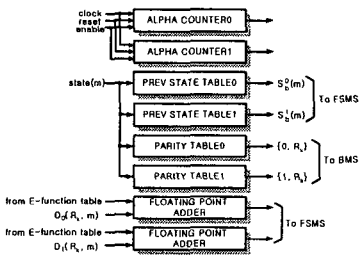

그림 3.4 FSMC 블록

2.1.5 LLRC(log likelihood ratio metric calculator) 블록

LLR(logarithm of likelihood ratio)를 구하기 위한 LLRC블록을 나타내면 그림 3.5과 같이 구성할 수 있다. LLRC 카운터를 이용하여 k개의 LLR을 계산하게 된다. 수식에 사용되는 상태 값은 테이블을 이용하여 계산하게 되며 메모리의 어드레스로 사용된다. 메모리에서 역방향 메트릭값을 참조하게 되며 순방향 메트릭과 계산한 결과 값을 E 함수 테이블에 보내고 참조된 값을 입력으로 받아 들여 LLR을 계산하게 된다. 여기서 LLRC 블록이 RSMC나 FSMC 블록과 다른 점은 수식에서 볼 수 있듯이 E 함수 테이블을 상태 수만큼 참조해서 그 때의 LLR을 계산하게 된다는 점이다.

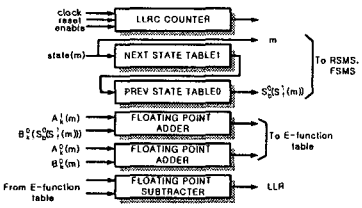

그림 3.5 LLRC 블록

2.2 시뮬레이션 결과 및 복잡도 분석

2.2.1 Simulation 결과

그림 4.1는 부호율이 1/2이고 (23,35) RSC 부호기를 사용했을 때 AWGN 채널상에서의 C 시뮬레이션에서의 BER과 설계한 MAP 복호기에서의 BER성능을 비교한 그래프이다. 시뮬레이션 결과는 MAP 복호기 하나일 때의 성능이며 터보 디코더에 적용하여 여러 번의 반복을 수행하면 더 좋은 성능을 얻을 수 있을 것이다. 본 논문에서는 MAP 복호기의 구현을 부동소수점연산을 이용하였고 C++에서의 시뮬레이션결과와 거의 유사한 결과를 얻을 수 있었다. 시뮬레이션 결과에서 동그라미표가 되어있는 선이 설계된 MAP복호기의 BER 그래프를 나타내고 있으며 네모 표가 되어있는 선이 C++ 시뮬레이션에서의 BER 그래프를 나타낸 것이다.

2.3.2 복잡도 분석

표 4.1에서 보는 바와 같이 Log-MAP 알고리즘의 사용으로 모든 곱셈의 계산이 덧셈으로 계산되어지며 logarithm 과 exponential의 계산은 E함수를 이용하여 계산하여 복잡도를 줄일 수 있다. 또한 MAP 복호기에서는 많은 메모리의 사용이 필요한데 본 논문에서 사용한 복호 알고리즘은 δ을 저장하기 위한 2^nN (여기서 n = 2이고 블록의 크기가 N인 경우)크기의 메모리와 β을 저장하기 위한 $2^{\nu-1}N$(ν는 부호기의 레지스터 크기)크기의 메모리가 필요하다.

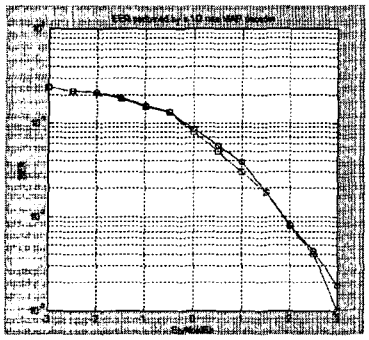

그림 4.1 설계된 MAP 복호기와의 BER성능 비교

	Pure-MAP	Log-MAP
δ	N×(2×2^{a})×(exponential and multiplications)	N×(2×2^{a})×(additions)
α	N×2^{a}×(1 addition and 2 multiplications)	N×2^{a}×(1 E function call and 2 additions)
β	N×2^{a}×(1 addition and 2 multiplications)	N×2^{a}×(1 E function call and 2 additions)
LLR	N×(1 logarithm and divisions and 2 × (2^{a}-1)additions and 2 × 2^{a} multiplications)	N×(1 subtraction and 2 × (2^{a}-1) E function calls and 2 × 2^{a} additions)

표 4.1 Pure-MAP 알고리즘과 Log-MAP 알고리즘의 복잡도 분석

3. 결론

본 논문에서는 MAP(Maximum A Posteriori) 복호 알고리즘을 이용한 MAP Decoder를 설계하였다. MAP 복호 알고리즘을 하드웨어로 구현하기 위하여 LOG-MAP 복호 알고리즘을 이용하여 MAP 복호기의 구조를 제시하고 동작원리에 관해서 설명하였다. MAP Decoder는 Turbo Decoder의 반복 복호에 적용할 수 있으며 연판정 입/출력(soft-in/soft-out)이 가능하다. Turbo Decoder에 사용되는 MAP 복호기는 Verilog HDL을 이용하였으며 설계한 후 Cadence Verilog-XL^{TM} 시뮬레이터로 동작을 검증하였다. 설계된 MAP 복호기는 Turbo Decoder의 반복 복호에 응용이 가능하다.

참고문헌

[1] B. Sklar, "Digital Communications -- Fundamentals and Applications", *Prentice Hall*, Chapter 5, 1988

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding Turbo codes," *Proc. Int. Conf. Comm.*, pp. 1064-1070, 1993

[3] C. Berrou and A. Glavieux, "Hear optimum error correcting coding and decoding: turbo-codes, " *IEEE Trans. Comm.*, pp. 1261-1271, Oct. 1996

[4] 김수영, 이수인, "터보 코드 개발동향", 주간기술동향 통권 888호, 한국전자통신연구원, 1999.3.13

[5] Hagenauer, P. Roberston and L. Papke, "Iterative (turbo) decoding of systematic convolutional codes with the MAP and SOVA algorithms," *ITG Conf., Frankfurt, Germany*, Oct. 1994

[6] S. S. Pietrobon, "Implementation and performance of a serial MAP decoder for use in an iterative turbo decoder, " *IEEE Int. Symp. Inform. Theory & its Applic., Sydney, Australia*, pp. 1073-1077, Nov. 1994

[7] Bahl. L. R., Cocke, J., Jeinek, F. and Raviv, J., "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Transactions on Information Theory, Vol. IT-20*, pp. 248-287, 1974

[8] S. S. Pietrobon and A. S. Barbulescu, "A simplification of the modified Bahl decoding algorithm for systematic convolutional codes, " *Int. Symp. Inform. Theory & its Applic., Sydney, Australia*, pp. 1073-1077, Nov. 1994

[9] Forney, G. D., "The Viterbi Algorithm", *Proceedings of IEEE, Vol. 61.*, pp. 268-278, 1973

[10] 김수영, 홍성원, "터보 코드 연구동향", 주간기술동향 통권 843호, 한국전자통신연구원, 1998.4.23.