소프트웨어 기반 채널 코딩을 위한 Integer 부호의 소개 및 성능 분석

하태욱, 나희주, 공동현, 김상효*
성균관대학교 전자전기컴퓨터공학과

*iamshkim@skku.edu

On the Integer codes for S/W based channel coding

Taeuk Ha, Hee Ju Na, Dong Hyun Kong, Sang-Hyo Kim*
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

오늘날 유한체(Finite field)에서의 연산을 기반으로 설계된 많은 수의 오류정정부호(Error Correction Code, ECC)들이 채널 코딩에 사용되고 있지만 이러한 부호들은 유한체 연산을 위한 추가적인 하드웨어를 필요로 한다. Integer 부호는 정수환(Integer ring)에서의 연산을 기반으로 설계된 오류정정부호로 부호화 및 복호화 과정에서 정수의 사칙연산과 LUT (Look Up Table) 연산만을 이용하기 때문에 GPP (General Purpose Processor)에서 하드웨어의 추가 없이 소프트웨어로 구현하기에 적합하다. 본 논문은 Integer 부호에 대해 소개하고 유한체 연산 기반 부호들과 오류정정성능을 비교한다.

Ⅰ. 서 론

1948년 Shannon의 channel capacity theorem[1] 이후, 채널 용량에 도달하는 부호를 찾기 위해 많은 연구가 이루어졌고 오늘날 CRC (Cyclical Redundancy Check) 부호, BCH (Bose-Chaudhuri-Hocquenghem)[2] 부호, RS (Reed-Solomon)[3] 부호 등의 부호들이 통신 과정에서 발생하는 오류를 제어하기 위해 사용되고 있다. 하지만 이 부호들은 유한체에서의 연산을 기반으로 설계되기 때문에 부호화 및 복호를 위해선 유한체 연산을 위한 회로가 포함된 하드웨어를 필요로 한다[4].

정수환에서의 연산을 이용하여 Integer 부호를 설계하는 방법은 1972년 처음 제안되었지만 이진 연산을 기반으로 설계하는 부호들에 비해 낮은 오류정정능력 때문에 주목받지 못하였다[5,6]. 그러나 Integer 부호는 부호화 및 복호 과정에서 정수의 사칙연산과 LUT (Look Up Table) 연산만을 이용하기 때문에 하드웨어의 추가 없이 소프트웨어로 구현이 용이하다는 점은 초고속 데이터 처리 구현에 있어 주목할만한 특징이다.

본 논문에서는 Integer 부호의 특징과 설계 방법을 소개하고 그 활용 가능성을 확인하고자 논문 [4]에서 제안된 SEC-DEC-TAEC$_1$(Single Error Correction-Double Error Correction-Triple Adjacent Error in one symbol Correction) Integer 부호와 RS 부호, BCH 부호의 성능을 비교한다.

Ⅱ. 본 론

A. Integer 부호

Integer 부호는 b bit 길이의 심볼 k개를 메시지로 갖고 1개 심볼을 패리티로 사용하는 오류정정부호이다[4]. 패리티 심볼 B_{k+1}을 계산하는 데에는 메시지 심볼들의 정수 값 B_i와 계수 $C_i\ (1 \le i \le k)$ 를 필요로 한다. 심볼의 정수 값 $B_i = \sum_{n=0}^{b-1} a_n 2^n, a_n \in \{0,1\}$ 는 한 심볼의 b개 비트를 정수로 표현한 값을 의미하고, 계수 C_i는 목표로 하는 오류정정 능력에 따라 특정한 값을 갖는다. 이때 패리티 심볼은 $B_{k+1} = \sum_{i=1}^{k} C_i B_i \pmod{2^b - 1}$ 으로 계산된다. 이 패리티 심볼은 k 개의 메시지 심볼 뒤에 붙어 $k+1$ 길이의 부호어 $\mathbf{x} = (B_1, B_2, \cdots, B_k, B_{k+1})$ 가 만들어지고, 이 부호어는 채널을 통과하는 과정에서 잡음이 더해져 수신단에선 오류를 포함한 데이터 $\mathbf{y} = (\underline{B_1}, \underline{B_2}, \cdots, \underline{B_k}, \underline{B_{k+1}})$을 수신한다. 오류를 정정하기 위해선 수신 데이터로부터 신드롬 $S = \sum_{i=1}^{k} C_i \underline{B_i} - \underline{B_{k+1}} \pmod{2^b - 1}$을 계산하는데, 심볼 값의 오류를 $E_i = \underline{B_1} - B_1$ 라고 하면 $S = \sum_{i=1}^{k} C_i E_i - E_{k+1} \pmod{2^b - 1}$ 가 되어 메시지에 관계 없이 오직 오류에 의해 신드롬 값이 결정된다. 신드롬을 계산한 뒤 해당하는 심볼 오류 값과 위치를 신드롬 테이블에서 LUT연산을 통해 구하고, 이를 수신 데이터 $\mathbf{y}$에 더해 오류를 정정한다.

Integer 부호는 다른 부호와 구분되는 여러 특징을 갖고 있는데, 그중 하나는 정정하고자 하는 서로 다른 오류 패턴이 동일한 신드롬 값을 갖는다는 점이다. 예를 들어 한 심볼에 11

$$\text{Scenario1: } B_1 = 00000000000_2 = 0_{10} \rightarrow \underline{B_1} = 00000000\underline{1}00_2 = 4_{10}$$

bit가 들어간다고 할 때, 다음과 같이 첫번째 심볼의 9번째 비트에서 오류가 발생해 원래 0이었던 비트가 1로 뒤집힌 경우

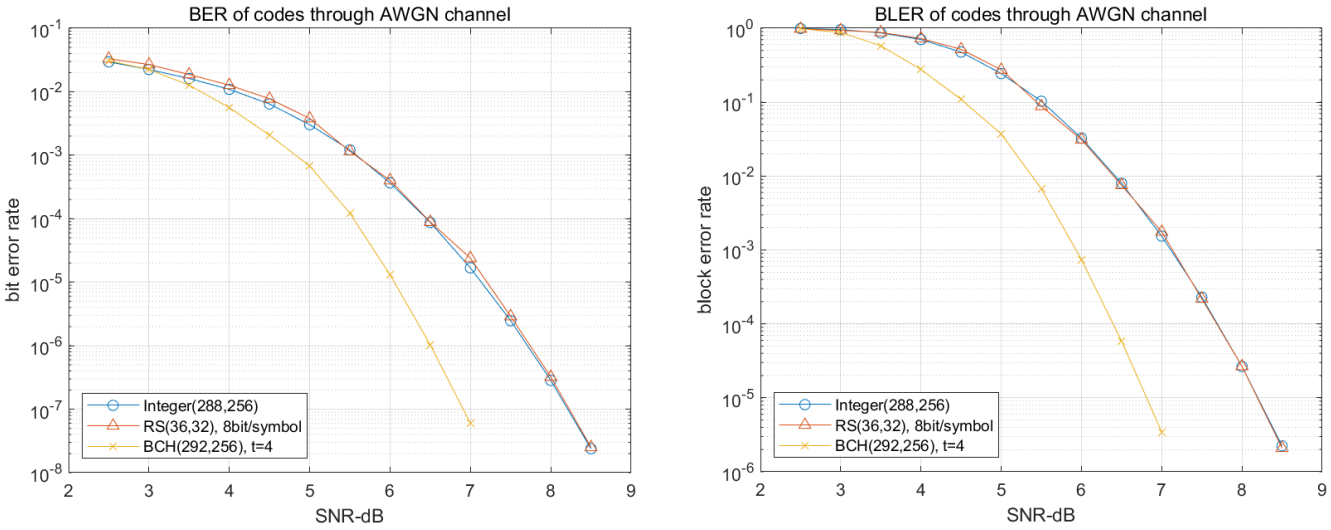
$$\text{Scenario2: } B_1 = 00000000100_2 = 4_{10} \rightarrow \underline{B_1} = 00000001\underline{0}00_2 = 8_{10}$$

신드롬 값은 $4C_1$이 된다. 그런데 Scenario2 와 같이 첫번째 심볼의 8번째 심볼이 0에서 1로 뒤집히고, 9번째 심볼이 1에서 0으로 뒤집히는 경우에도 신드롬 값은 $4C_1$을 갖게 된다. 그렇지만 두 경우 모두 비트 오류로 인해 더해지는 오류의 정수 값이 동일하기 때문에 정정을 위해 더해주는 값 또한 동일하여 하나의 신드롬 값으로 두 오류 패턴을 정정할 수 있다. 정정 목표 오류 패턴들의 신드롬 집합 사이 이러한 포함 관계를 이용해 계수 C_i의 조건을 얻을 수 있다.

또 다른 중요한 특징은 Integer 부호는 그 부호화와 복호 과정에서 정수의 사칙연산과 LUT 연산만을 이용한다는 점이다. 이는 보편적인 프로세서에서 수행할 수 있는 연산이기 때문에 유한체 연산을 위한 추가적인 하드웨어를 필요로 하지 않고, 또한 그렇기 때문에 소프트웨어로의 구현이 용이하다는 특징이 있다.

B. 시스템 모델 및 오류 채널

전송하고자 하는 메시지 $\mathbf{m} \in \{0,1\}^k$ 의 통신 과정에서 발생할 수 있는 오류로 부터 보호하기 위해 메시지를 인코더에 통과시켜 패리티 비트가 추가된 부호어 $\mathbf{x} \in \{0,1\}^n$를 만든다. 부호어는 채널을 통해 수신단으로 송신되고, 이 과정에서 잡음에 의해 오류 $\mathbf{e}$가 더해져 수신 데이터 $\mathbf{y} = \mathbf{x} + \mathbf{e}$ 를 수신하게 된다. 수신한 $\mathbf{y}$ 를 디코더에 통과시켜 오류를 정정하고 예상되는 부호어 $\hat{\mathbf{x}}$ 및 메시지 $\hat{\mathbf{m}}$을 얻는다.



(a) BER 그래프 (b) BLER 그래프

그림 1. AWGN 채널에서 Integer 부호, RS 부호, BCH 부호의 성능 그래프

부호어 $\mathbf{x}$가 통과하는 채널의 종류에 따라 더해지는 오류가 각기 다른데, 본 논문의 실험에선 무선 통신 환경에서의 사용을 가정해 AWGN (Additive White Gaussian Noise) 채널을 적용하였다.

C. 성능 분석

256 bit 길이 메시지를 각각 Integer 부호, BCH 부호, 그리고 RS 부호를 이용해 부호화하고 이 부호어를 AWGN 채널에 통과시킨 후 복호하여 송신 부호어 $\mathbf{x}$ 와 복호 부호어 $\hat{\mathbf{x}}$를 비교, 그 결과를 통해 BER (bit error rate)과 BLER (block error rate)을 계산하였다. 이를 채널의 SNR (Signal-to-Noise Ratio)을 바꿔가며 반복하여 유한체 연산을 기반으로 설계한 부호와 Integer 부호의 성능 차이를 확인하고자 하였다.

Integer 부호는 심볼 당 비트 수 $b = 32$, 메시지 심볼 수 $k = 8$ 인 SEC-DEC-TAEC$_1$ (288,256) Integer 부호를 이용하였다.

BCH 부호는 부호 길이 $n = 2^m - 1$일 때 최대 t 개의 비트 오류를 정정하기 위해 $r = m \cdot t$개의 패리티를 필요로 하는데, 세 부호 모두 유사한 정보율을 유지하면서 256 bit 메시지 길이에 맞추기 위해 $m = 9, t = 4$인 (511,475) BCH 부호를 219 bit 만큼 단축(shortening)하여 (292,256) 단축된 BCH 부호를 사용하였다.

RS 부호는 심볼 단위 오류정정부호로, 심볼 당 비트 수가 a일 때 부호 길이 $n = 2^a - 1$ 심볼이 되고, 최대 t개 심볼 오류를 정정하기 위해 $2t$개의 패리티 심볼을 필요로 한다. $a = 8$이고 $t = 2$인 RS 부호를 설계하면 (255,251) 심볼 길이의 RS 부호를 얻을 수 있는데, 이를 단축하여 (36,32) 단축된 RS 부호를 실험에 사용하였고, 이를 비트 수로 계산하면 Integer 부호와 동일한 (288,256)길이가 된다.

메시지 길이가 256 bit일 때 Integer 부호의 성능을 BCH 부호, RS 부호와 비교한 결과 Integer 부호는 RS 부호와 거의 동일하고 BCH 부호보다 약 100배 이상 높은 오류 확률을 보였는데, 이는 세 부호의 오류정정능력의 차이에 인한 결과로 실험에 사용된 BCH 부호는 최대 4개의 비트 오류를 정정할 수 있는 반면 RS 부호는 최대 2개의 8bit 심볼 오류 정정, Integer 부호는 SEC-DEC-TAEC$_1$ 오류정정 능력을 갖기 때문에 각 비트마다 독립적으로 오류가 발생하는 AWGN 채널에선 Integer 부호와 RS 부호의 성능이 거의 동일하고 BCH 부호가 뛰어난 성능을 보인다.

따라서 메시지 길이와 채널의 설정에 따라 결과가 바뀔 수 있는데, 채널에 버스트(burst) 오류 발생 확률이 존재할 경우 심볼 단위 오류 정정을 하는 RS 부호와 TAEC$_1$이 가능한 Integer 부호의 오류 확률이 BCH 부호보다 더 낮을 수 있다. 하지만 이 경우에도 RS 부호의 버스트 오류 정정 능력이 더 뛰어나기 때문에 Integer 부호는 RS 부호보다 높고 BCH 부호보다 낮은 오류 확률을 보일 것으로 예상되며, 이는 추가 실험을 통해 확인해야 한다.

Ⅲ. 결 론

Integer 부호는 RS 부호, BCH 부호와 달리 GPP (General Purpose Processor)에서 수행 가능한 연산만을 이용하기 때문에 유한체 연산을 위한 하드웨어의 추가 없이 RS 부호 혹은 BCH 부호와 동일한 성능을 가질 수 있으며 소프트웨어를 이용한 채널 코딩 구현에 적합하다. Integer 부호의 이러한 장점을 이용할 수 있는 환경과 해당 환경에서의 오류 채널 모델 그리고 그에 알맞은 오류정정 능력을 갖는 Integer codes의 설계에 대해 추가적인 연구를 진행해야 한다.

ACKNOWLEDGMENT

이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2021-0-00863, 고신뢰 메모리를 위한 지능형 인메모리 오류정정 디바이스 개발).

참 고 문 헌

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. Journal,* vol. 27, no. 3, pp. 379-423, July 1948.

[2] R. Bose and D. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Inform. Contr.*, vol. 3, no. 1, pp. 68 - 79, 1960.

[3] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300-304, Jan. 1960.

[4] A. Radonjic, "Integer Codes Correcting Double Errors and Triple-Adjacent Errors Within a Byte," *IEEE Trans. VLSI Systems*, vol. 28, no. 8, pp. 1901-1908, Aug. 2020.

[5] I. F. Blake, “Codes over certain rings,” *Inform. Contr.*, vol. 20, pp. 396-404, 1972.

[6] A. Radonjic and V. Vujicic, "Integer Codes Correcting Burst Errors within a Byte," *IEEE Trans. Computers,* vol. 62, no. 2, pp. 411-415, Feb. 2013.