비트 차원의 치환을 기반으로 한 극 부호 부분 병렬 부호기

신예린, 최소연, 유호영
충남대학교 전자공학과

Partially Parallel Encoder Architecture for Polar Codes based on Bit-dimension Permutations

Yerin Shin, Soyeon Choi, and Hoyoung Yoo
Chungnam National University

Abstract - 극 부호의 채널용량 달성 특성은 극 부호를 각광 받는 오류 정정 부호로 만들었다. 하지만 충분한 오류 정정 성능은 부호의 길이가 길어졌을 때 달성되는 점근적 속성을 보인다. 따라서 입력 데이터가 길어지는 경우에 대한 초대규모 집적회로 구현을 실현하기 위하여 효율적인 구조가 필요하게 되었다. 기존의 극 부호 부호기 구조 중 가장 기본적인 완전 병렬 부호기는 직관적이고 구현하기 쉽지만 긴 극 부호에 높은 하드웨어 복잡성을 보이므로 적합하지 않다. 그리고 이를 보완하여 제안된 부분 병렬 부호기는 하드웨어 면적 측면에서 큰 성과를 얻었으나 그 방식이 일반화되어 있지 않고 설계자에 따라 구조가 다르게 나타날 수 있다. 본 논문에서는 이를 개선하고자 최적화된 비트 차원의 치환을 위해 제안된 회로 설계법을 극 부호에 적용하여 완전 병렬 구조 만큼 직관적인 구조를 제시한다. 제안하는 구조를 극 부호의 부호기에 적용하면 부호 길이가 1024이고 병렬 차원의 크기가 8일 때 하드웨어 상에서 완전 병렬 부호기보다 1.85배 적은 면적을 차지한다.

1. 서 론

극 부호 (polar code)는 상대적으로 낮은 복잡도로 채널용량이 달성될 수 있음이 증명된 새로운 종류의 오류 정정 코드이다[1]. 이러한 특성은 극 부호를 부호이론의 돌파구로 간주하기에 충분하였으며 그 결과로 데이터 저장 장치를 비롯한 많은 분야에서 그 적용 가능성이 연구되고 있다.

하지만 극 부호가 채널용량을 달성하기 위해서는 부호의 길이가 충분히 길어야 한다. 부호의 길이가 길어지는 경우 극 부호의 부호화와 복호화 과정에 필요한 하드웨어의 복잡도가 증가하고 지연 시간 또한 길어진다. 따라서 부호의 길이가 긴 극 부호를 효율적으로 처리할 수 있는 하드웨어 구조의 필요성이 대두되었다.

극 부호의 대표적인 복호 방법인 연속-제거 (Successive cancelation; SC) 복호[2] 그리고 신뢰-전파 (belief propagation; BP) 복호[3]와 같이 복호 방법들에 관한 연구는 활발히 진행되고 있다. 하지만 상대적으로 극 부호의 부호화를 위한 하드웨어 구조에 관한 연구는 활발하게 진행되지 않았다. 가장 먼저 모든 메시지 비트를 완전 병렬적 방식으로 처리하는 간단한 부호기 구조가 제시되었다 [1]. 이 방법은 직관적이고 구현이 쉽다는 장점이 있지만, 모든 부호를 동시에 처리해야 하므로 면적 측면에서 적합한 방법이라 할 수 없다. 그 후 완전 병렬 구조의 단점을 보완한 부분 병렬 구조가 제안되었다 [4]. 이 구조는 완전 병렬 구조의 부호기에 비해 하드웨어 면적을 절약할 수 있다는 장점이 있지만, 레지스터 할당 방법에 따라 하드웨어 구조에 변동성을 가진다. 본 논문에서는 비트 차원의 치환(bit-dimension permutation)을 위해 제안된 하드웨어 설계법 [5]를 적용하여 확실한 규칙성으로 일반화된 극 부호 부호기 설계 기법을 제시한다.

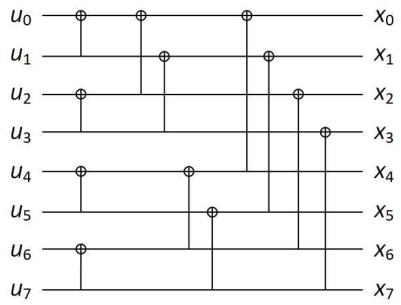
2. 본 론

2.1 극 부호 부호화 (Polar Encoding)

극 부호는 N개의 동일한 서브 채널 집합으로 결합 된 채널로부터 부호의 길이가 무한대에 가까워질수록 각 채널이 완전한 신뢰성을 보이거나 혹은 완벽히 신뢰성을 잃게 되는 채널 양극화 현상을 이용한다 [1]. 각 서브 채널의 신뢰성은 선험적으로 알 수 있으므로 가장 신뢰할 수 있는 서브 채널 K개를 사용하여 정보를 전송하고 나머지 서브 채널은 미리 결정한 값으로 설정하여 (N,K) 극 부호를 구성한다.

극 부호는 선형 블록 코드(linear block code)로 부호화 과정은 생성 행렬 (generator matrix) G에 의해 특징지어진다. 길이가 N 또는 2^n인 부호에 대한 생성 행렬 G는 커널 행렬 $F=\begin{bmatrix}1&0\\1&1\end{bmatrix}$를 크로네커 거듭제곱하여 얻을 수 있다. 부호어 (codeword)는 $x=uG_N$으로 구해지며, 이때 u와 x는 각각 유효한 정보와 부호어를 나타낸다. 극 부호의 부호화를 위한 완전 병렬 부호기 구조가 [1]에 제시되었고 길이 N의 극 부호에 대해 $O(N\log N)$의 부호화 복잡도를 가지며 $N=2^n$ 일 때 n단계로 나뉘어 부호화 과정이 진행된다.

그림 1에 N 이 8로 3단계에 걸쳐 12개의 XOR 게이트로 구현되는 완전 병렬 구조 부호기를 나타내었다. 완전 병렬 부호기는 생성 행렬 G를 기반으로 직관적으로 설계되며 전체 부호화 과정이 한 사이클에 완료된다. 하지만 실제 구현에서 극 부호의 부호화를 위해 완전 병렬 구조를 택하는 것은 부호 길이가 길어짐에 따라 메모리 크기와 XOR 게이트의 수가 증가하기 때문에 면적 측면에서 비효율적이다. 이에 대한 대안으로 제안된 폴딩



<그림 1> 16bit 극 부호의 부호화를 위한 완전 병렬 구조

(Folding) 구조가 적용된 부분 병렬 방법 [4] 은 유도되는 과정이 복잡할 뿐만 아니라 설계자의 의견이 설계 결과에 영향을 미쳐 하드웨어의 면적과 지연 시간 등 성능적인 측면에서 차이가 발생할 수 있다. 본 논문에서는 최적화된 비트 차원의 치환 하드웨어 설계 기법 [5] 을 극 부호의 부분 병렬 부호기 설계에 적용하여 일관된 규칙성을 확보하고 완전 병렬 구조의 부호기보다 효율을 높였다.

2.2 비트 차원의 치환(Bit-dimension Permutations)

비트 차원의 치환이란 n 차원의 공간 속에 있는 데이터를 치환 좌표에 따라서 재배열하는 것을 의미한다. 이를 수행하는 최적의 하드웨어를 설계하기 위한 체계적인 방법이 [5] 에 제시되었다. 제안된 방법의 기본 원리는 비트 치환을 기본적인 비트의 교환으로 분해하여 보는 것이다. 기본적인 비트의 교환 (Elementary Bit Exchange, EBE)은 최종적으로 x_j 그리고 x_k 두 차원 사이에서 일어난다.

이 방법을 사용하기 위하여 데이터의 흐름을 모델링 하는 과정이 필요하다. 데이터의 흐름은 관습적으로 왼쪽에서 오른쪽으로 진행되며 크게 직렬 차원 (serial dimension)과 병렬 차원 (parallel dimension)으로 구분하여 볼 수 있다. 직렬 차원에서는 다른 클락 사이클에 같은 터미널을 통하여 데이터가 이동하고 병렬 차원에서는 다른 터미널을 통해 같은 시간에 데이터가 전달되게 된다. 결과적으로 데이터 흐름 모식도는 직렬로 2^{n-p} , 병렬로 2^p 개의 데이터로 구성된 직사각형 모양을 가진다. 이 직사각형의 공간 속에서 각 데이터의 위치 P 가 정의된다. n 차원의 공간에서 각 데이터의 위치 값은 0에서 2^n-1까지의 번호로 식(1)에 의해 매겨진다.

$$P=\sum_{i=0}^{n-1} x_i\ 2^i. \tag{1}$$

그리고 모식도에서 도착 시간 (time of arrival) t와 데이터의 터미널 T를 어떠한 위치에서든 정의할 수 있다. 도착 시간 t는 회로 안의 주어진 지점에서 첫 번째 표본이 도착하는데 걸리는 시간을 의미하며 오로지 직렬 차원에만 의존적인 값이다.

$$t(P)=\sum_{i=0}^{n-1} x_i\ 2^{i-p}. \tag{2}$$

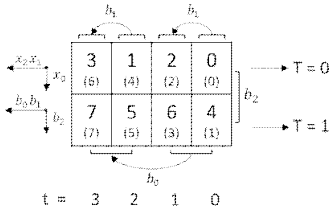
터미널 T는 식(3)을 통해 정의되며 도착 시간 t와는 반대로 오직 병렬 차원에만 의존적이다.

$$T(P)=\sum_{i=0}^{p-1} x_i\ 2^i. \tag{3}$$

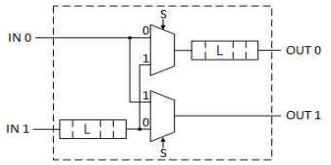
이 두 값을 사용하여 위치 P를 다시 나타낼 수 있다.

$$P=t\ |\ T. \tag{4}$$

이때 바(bar) |는 직렬 차원과 병렬 차원을 구분하는 표시이다. 또한, 신호처리 알고리즘에서는 인덱스 된 데이터들로 수학 연산을 정의하기 때문에 데이터를 인덱스화 하여야 하며, b_i는 인덱스의 10진수 값이고 2^i는 이진 표현의 비트이다.



〈그림 2〉 $p \equiv b_o b_1 | b_2$ 을 나타낸 데이터 흐름 모식도



〈그림 3〉 직렬-직렬 차원의 치환을 위한 기본적인 회로

$$I = \sum_{i=0}^{n-1} b_i \, 2^i \quad . \tag{5}$$

또한, P를 I의 함수로 나타낼 수 있으며, 인덱스화 된 데이터가 데이터 흐름의 각 위치에 할당되게 된다.

그림 2는 구한 모든 값을 종합하여 하나의 병렬 차원과 두 개의 직렬 차원을 가지는 데이터 흐름을 나타낸 것이다. 연관된 도착 시간 t 와 터미널 T를 표시하고 위치 P 괄호 안에, 데이터의 인덱스 값은 직사각형 공간 안에 괄호 없이 표기하였다.

2.3 비트 차원의 치환을 기반으로 한 극 부호 부분 병렬 부호기 구조

비트 차원의 치환을 위한 극 부호 부분 병렬 부호기 설계는 비트 차원의 치환이 일련의 기본적인 비트의 교환으로 분해된다는 원칙을 기반으로 한다 [5]. 이 원칙에 따라 하드웨어 구조를 크게 세 가지의 경우로 나누어 볼 수 있다. 본 논문에서는 극 부호에서 관찰될 수 있는 두 경우에 대해서만 간략히 살펴보도록 한다. 공통으로 데이터의 입력 순서를 p_0 출력 순서를 p_1이라 하고 와 가 모두 병렬 차원에 존재하며 $p > j > k$ 이라고 가정한다.

2.3.1 두 병렬 차원 간의 기본적인 비트 교환

$$p_0 \equiv u_{n-1} \ \cdots u_p \, | u_{p-1} \ \cdots u_j \ \cdots u_k \ \cdots u_0$$
$$p_1 \equiv u_{n-1} \ \cdots u_p \, | u_{p-1} \ \cdots u_k \ \cdots u_j \ \cdots u_0 \ . \tag{5}$$

두 병렬 차원 간의 치환에는 직렬 차원이 포함되지 않으므로 같은 시간에 다른 터미널을 통해 데이터가 전달되며, 식 (5)는 치환 과정을 식으로 나타낸 것이다. 따라서 단순히 두 입력 터미널과 출력 터미널을 상호 연결해 줌으로써 비트 치환을 완료할 수 있다.

2.3.2 직렬, 병렬 차원 간의 기본적인 비트 교환

$$p_0 \equiv u_{n-1} \ \cdots u_j \ \cdots u_p | u_{p-1} \ \cdots u_k \ \cdots u_0$$
$$p_1 \equiv u_{n-1} \ \cdots u_k \ \cdots u_p | u_{p-1} \ \cdots u_j \ \cdots u_0 \ . \tag{6}$$

직렬, 병렬 차원 간의 치환에는 치환되어야 하는 입력 데이터 쌍이 다른 터미널을 통해 전달되며, 치환 과정을 식으로 나타내면 식 (6)과 같다. 다른 시간에 다른 터미널을 통해 치환되어야 하는 데이터가 입력되므로 데이터의 도착 소요 시간에 차이가 발생한다.

$$\triangle t = 2^{j-p} = L. \tag{7}$$

이는 치환을 위해 요구되는 최소한의 지연 시간, L과 동일하게 된다. 이 지연 시간의 영향을 반영하기 위해 L과 같은 길이의 버퍼 한 쌍과 치환을 위한 MUX 2개로 기본적인 셔플링 회로인 그림 3이 설계된다.

두 MUX는 같은 컨트롤 시그널에 의해 통제되며 병렬 차원이 하나 이상일 경우, 그림 3의 회로가 2^{p-1}번 중복되어 결과 출력까지 식(8)의 딜레이를 필요로 한다.

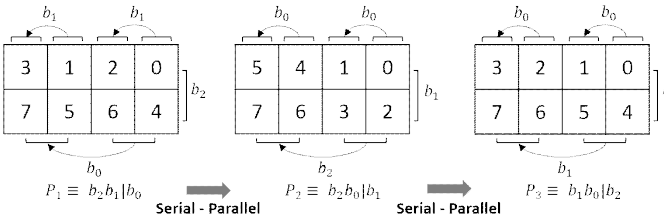
$$D(\sigma) = 2^{p-1} \cdot 2L = 2^j. \tag{8}$$

이를 종합하여 전체 과정을 사이클로 구분하고 각 사이클마다 경우에 맞는 기초 회로를 적용해 비트 치환을 위해 최적화된 하드웨어 회로를 일반화하여 구할 수 있다.

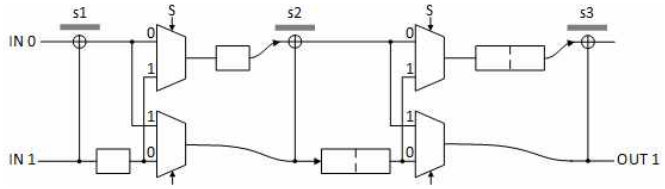
2.4 극 부호를 위해 일반화된 부분 병렬 인코더

극 부호의 부호화를 위하여 거쳐야 하는 일련의 단계들 사이에는 치환이 요구된다. 부호화를 위한 치환이 필요한 부분에 비트 차원의 치환을 적용하여 극 부호 부분 병렬 부호기를 설계할 수 있다. 그림 4에 부호의 길이 N이 8이면서 병렬 차원이 1인 경우에 대한 데이터 흐름 모식도를 데이터 크기에 따라 3단계로 나타내었으며, 인덱스화 한 값과 단계가 전환될 때 이루어져야 하는 치환의 종류를 표기하였다.

그림 4의 데이터 흐름 모식도를 바탕으로 하여 극 부호 부호기에 비트



〈그림 4〉 N=8일 때, 단계별 극 부호 부호기의 데이터 흐름 모식도



〈그림 5〉 비트 차원의 치환을 적용한 극 부호의 부분 병렬 부호기 구조

〈표 1〉 N 이 1024일 때 극 부호 인코더 합성 결과

Parallel-dimension	Area	Clock cycle
4	8,746.66	127
8	11,946.51	7
10	22,186.05	1

차원 치환을 적용한 회로를 그림 5에 제시하였다. 제안하는 비트 치환 회로를 적용한 극 부호 부분 병렬 부호기를 사용하면 부호의 길이를 늘이더라도 직관적으로 데이터의 흐름을 모델링 할 수 있다. 또한, 일반화되어 있는 회로를 적용하여 손쉽게 최적화된 극 부호 부호기를 설계할 수 있다.

3. 실험 결과 및 결론

극 부호의 부호 길이 N을 1024로 고정하고 병렬 차원의 크기를 각각 4, 8, 10으로 설정한 극 부호 부분 병렬 부호기를 구현하였다. 인코더의 동작 주파수를 동작 주파수를 200MHz로 가정하고 CMOS 180nm 공정을 이용하여 합성을 진행한 결과를 표 1에 정리하였다.

표 1에 의하면 병렬 차원이 완전 병렬 인코더와 동일한 구조를 보이는 10인 경우가 4인 경우에 비해 2.54배 더 큰 하드웨어 면적을 나타냈다. 하지만 이와는 반대로 최종 결과가 출력될 때까지 필요한 클락 사이클의 수는 병렬 차원이 늘어나 완전 병렬 구조에 가까워질수록 줄어든다. 즉 병렬 차원이 커질수록 하드웨어 면적은 줄어들지만, 결과 출력까지 소요되는 시간은 증가한다.

이처럼 결과가 출력되는데 걸리는 시간과 하드웨어 면적 간에는 트레이드오프가 존재하지만, 현재까지 극 부호를 효율적으로 부호화하는 부호기 하드웨어 설계 방법에 관한 연구는 복호기에 비해 이루어지지 않았다. 본 논문에서 제안하는 비트 차원 치환 기법을 적용한 극 부호 부호기 설계법은 기존의 부분 병렬 극 부호 부호기에 적용된 방법과 비교해 보았을 때, 설계법이 더 직관적이며 확실한 규칙성을 보인다. 따라서 비트 차원의 치환을 위해 제안된 하드웨어 설계법을 극 부호에 적용시키는 본 방법은 극 부호 인코더를 위한 실용적인 솔루션을 제공한다.

[참 고 문 헌]

[1] Arikan, Erdal. "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels." IEEE Transactions on information Theory 55.7 (2009):3051-3073.
[2]Yuan, Bo, and Keshab K. Parhi. "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding." IEEE Transactions on Circuits and Systems I: Regular Papers 61.4 (2013): 1241-1254.
[3]Arıkan, Erdal. "Polar codes: A pipelined implementation." Proc. 4th Int. Symp. on Broad. Commun. ISBC 2010. 2010.
[4] Yoo, Hoyoung, and In-Cheol Park. "Partially parallel encoder architecture for long polar codes." IEEE Transactions on Circuits and Systems II: Express Briefs 62.3 (2014): 306-310.
[5] Garrido, Mario, Jesús Grajal, and Oscar Gustafsson. "Optimum circuits for bit-dimension permutations." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27.5 (2019): 1148-1160.
[6]Fraser, Donald. "Array permutation by index-digit permutation." Journal of the ACM (JACM) 23.2 (1976): 298-309.