Linear Block Code

https://youtu.be/k20oFEZSvY0

Error Correction Code(ECC)

- 메시지를 이진수로 전송하였을 때, 메시지의 일부 비트가 잘못 뒤집히더라도 메시지를 복구 할 수 있는 인코딩 방식
  - 실제로 0인 비트가 1로 전송되거나 그 반대의 경우

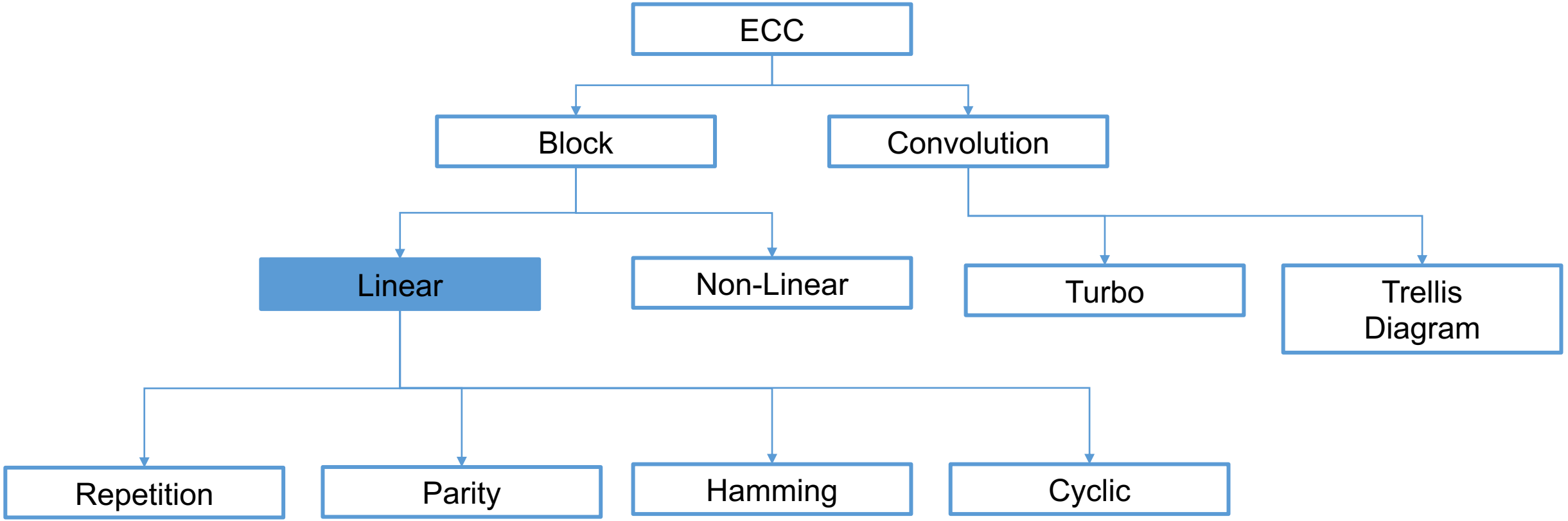
- 데이터 손상을 방지가 필요한 경우 대부분 사용

- 제일 쉬운 ECC의 예시
  - 비트가 여러 번(3회 이상) 전송된다고 할 때, 더 많이 전송된 비트를 올바른 비트로 판단
  - 전송된 비트 중 하나가 오류(단일 비트 오류)일 때 메시지 수정이 가능
  - 하지만, 기존의 3배 길이인 메시지를 전송 해야 하기 때문에 매우 비효율적
    - 동일한 메시지를 3회 이상 보내야 하기 때문

Received Message	Interpreted bit
000	0
110	1
010	0

https://brilliant.org/wiki/error-correcting-codes/

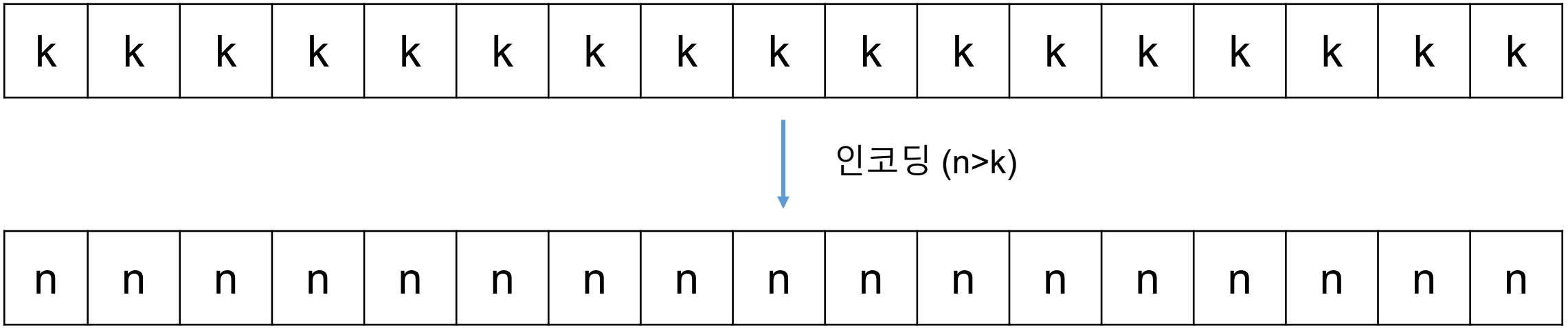
Error Correction Code(ECC)



https://www.micron.com/-/media/client/global/documents/products/technical-note/nand-flash/tn2963_ecc_in_slc_nand.pdf

Linear Block Codes

- 실제 메시지가 담긴 비트와 패리티 검사 비트를 선형적으로 결합
- Block Code에서의 완전한 메시지 비트
  - 각 블록이 동일한 수의 비트 보유
  - 각 블록이 k비트를 포함하고 블록의 k 비트가 Dataword 정의
    - 전체 Dataword : 2^k



Linear Block Codes

- 전체 Codeword = $2^n = 2^{17}$
- 전체 Dataword = $2^k = 2^{12}$
- Code rate(n, k) : $r_c = \frac{k}{n}$

https://electronicsdesk.com/linear-block-code.html

Linear Block Codes

- 1비트의 message symbol을 k 번 반복할 때,

1/k Repetition code로 표현

- 인코더 출력 n 이 3이라고 할 때,

가능한 Dataword = 2^3

* 단, Codeword에 위치한 임의의 2개의 값에 대하여 XOR 연산을 진행하였을 때,
그 결과값은 Cordword에 속해 있어야 함.

Dataword	Parity Bit	Codeword
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

Encoding Linear Block Codes

- (

Dataword	Parity Bit	Codeword
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

오.

$$d = [101], c = [1010]$$
$$c = dG$$
$$(G = \ generator\ matrix)$$

$$C = [101]\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

[항등부분행렬 + 패리티 부분행렬]

$$C = [1\ 0\ 1\ 0\ 1\ 1]$$

Encoding Linear Block Codes

MIT 6.02 **DRAFT** *Lecture Notes*
Last update: February 28, 2012
Comments, questions or bug reports?
Please contact hari at mit.edu

CHAPTER 6
Linear Block Codes: Encoding and Syndrome Decoding

The previous chapter defined some properties of linear block codes and discussed two examples of linear block codes (rectangular parity and the Hamming code), but the approaches presented for decoding them were specific to those codes. Here, we will describe a general strategy for encoding and decoding linear block codes. The decoding procedure we describe is **syndrome decoding**, which uses the syndrome bits introduced in the previous chapter. We will show how to perform syndrome decoding efficiently for any linear block code, highlighting the primary reason why linear (block) codes are attractive: the ability to decode them efficiently.

We also discuss how to use a linear block code that works over relatively small block sizes to protect a packet (or message) made up of a much larger number of bits. Finally, we discuss how to cope with burst error patterns, which are different from the BSC model assumed thus far. A packet protected with one or more coded blocks needs a way for the receiver to *detect* errors *after* the error correction steps have done their job, because all errors may not be corrected. This task is done by an *error detection code*, which is generally distinct from the correction code. For completeness, we describe the *cyclic redundancy check (CRC)*, a popular method for error detection.

6.1 Encoding Linear Block Codes

Recall that a linear block code takes k-bit message blocks and converts each such block into n-bit coded blocks. The rate of the code is k/n. The conversion in a linear block code involves only linear operations over the message bits to produce codewords. For concreteness, let's restrict ourselves to codes over $\mathbb{F}_2$, so all the linear operations are additive parity computations.

If the code is in systematic form, each codeword consists of the k message bits $D_1 D_2 \ldots D_k$ followed by the $n-k$ parity bits $P_1 P_2 \ldots P_{n-k}$, where each P_i is some linear combination of the D_i's.

Because the transformation from message bits to codewords is linear, one can represent

63

https://web.mit.edu/6.02/www/s2012/handouts/6.pdf

$$G = [\, I_k | P \,],$$

As a last example, suppose the parity equations for a $(6,3)$ linear block code are

$$\begin{aligned} P_1 &= D_1 + D_2 \\ P_2 &= D_2 + D_3 \\ P_3 &= D_3 + D_1 \end{aligned}$$

For this code,

$$G = \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right).$$

We denote the $k \times (n-k)$ sub-matrix of G by A. I.e.,

$$G = I_{k \times k} | A, \qquad (6.4)$$

Hamming Code

- 메시지 내에 중복 비트를 삽입하여 메시지 인코딩하는 방법
  - 지정된 위치에 비트 추가

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		...
	p2		✓	✓			✓	✓			✓	✓			✓	✓			✓	✓		
	p4				✓	✓	✓	✓					✓	✓	✓	✓					✓	
	p8								✓	✓	✓	✓	✓	✓	✓	✓						
	p16																✓	✓	✓	✓	✓	

  - 최대 2개의 동시 비트 오류 감지 가능
  - 단일 비트 오류에 대해서는 수정 가능

Hamming Code

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		...
	p2		✓	✓			✓	✓			✓	✓			✓	✓			✓	✓		
	p4				✓	✓	✓	✓					✓	✓	✓	✓					✓	
	p8								✓	✓	✓	✓	✓	✓	✓	✓						
	p16																✓	✓	✓	✓	✓	

Parity bit coverage	4	2	1
	0100 = 4	0010 = 2	0001 = 1
	0101 = 5	0011 = 3	0011 = 3
	0110 = 6	0110 = 6	0101 = 5
	0111 = 7	0111 = 7	0111 = 7
	...	...	...

Q & A