(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2009-0069127
(43) 공개일자 2009년06월29일

(51) Int. Cl.
H03M 13/27 (2006.01) *H04B 7/26* (2006.01)
H03M 13/00 (2006.01)
(21) 출원번호 10-2008-0074682
(22) 출원일자 2008년07월30일
심사청구일자 없음
(30) 우선권주장
61/016,492 2007년12월24일 미국(US)
(뒷면에 계속)

(71) 출원인
엘지전자 주식회사
서울특별시 영등포구 여의도동 20번지
(72) 발명자
노동욱
경기 안양시 동안구 호계동 533번지 LG제1연구단지
김기준
경기 안양시 동안구 호계동 533번지 LG제1연구단지
(뒷면에 계속)
(74) 대리인
김용인, 박영복

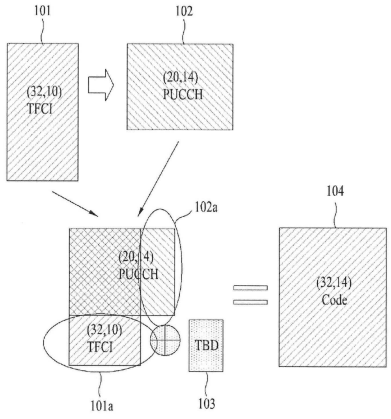
전체 청구항 수 : 총 14 항

(54) 블록 코드를 이용한 다양한 길이를 가진 정보의 채널 코딩방법

(57) 요 약

블록 코드를 이용한 다양한 길이를 가진 정보의 채널 코딩 방법이 개시된다. 구체적으로 32개의 행(Row) 및 정보 비트 길이에 해당하는 A개의 열(Column)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩함에 있어서, A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 채널 코딩된 결과를 출력 시퀀스로서 출력하며, 여기서 A가 10보다 큰 경우, 상기 코드 생성 행렬은 TFCI(Transport Format Combination Indicator) 정보의 코딩에 사용된 32개의 행 및 10개의 열로 구성된 TFCI 코드 생성 행렬에 대응하는 제 1 행렬 또는 이 제 1 행렬의 하나 이상의 행들 사이의 위치가 바뀐 제 2 행렬에 "0"을 10개 포함하는 조건을 만족하는 추가 기본 시퀀스들 중 A-10개의 추가 기본 시퀀스를 열 방향 시퀀스로서 추가한 것을 이용한다.

대 표 도 - 도1



(72) 발명자

안준기

경기 안양시 동안구 호계동 533번지 LG제1연구단지

이대원

경기 안양시 동안구 호계동 533번지 LG제1연구단지

노유진

경기 안양시 동안구 호계동 533번지 LG제1연구단지

(30) 우선권주장

61/021,337 2008년01월16일 미국(US)

61/028,016 2008년02월12일 미국(US)

특허청구의 범위

청구항 1

32개의 행(Row) 및 정보 비트 길이에 해당하는 A개의 열(Coulmn)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩하는 방법에 있어서,

상기 A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 상기 채널 코딩된 결과를 출력 시퀀스로서 출력하는 단계를 포함하며,

상기 A가 10보다 큰 경우, 상기 코드 생성 행렬은 TFCI(Transport Format Combination Indicator) 정보의 코딩에 사용된 32개의 행 및 10개의 열로 구성된 TFCI 코드 생성 행렬에 대응하는 제 1 행렬 또는 상기 제 1 행렬의 하나 이상의 행들 사이의 위치가 바뀐 제 2 행렬에 최소 해밍 거리의 최대값이 10인 조건을 만족하는 추가 기본 시퀀스들 중 A-10개의 추가 기본 시퀀스를 열 방향 시퀀스로서 추가한 것인, 채널 코딩 방법.

청구항 2

제 1 항에 있어서,

상기 추가 기본 시퀀스는 0을 10개 포함하는 기본 시퀀스인, 채널 코딩 방법.

청구항 3

제 1 항에 있어서,

상기 제 2 행렬은,

상기 제 2 행렬의 하단 12행을 제외한 제 3 행렬이 물리상향링크 제어 채널(PUCCH) 전송을 위한 코드 생성 행렬에 대응하도록 상기 제 1 행렬의 하나 이상의 행들 사이의 위치를 변경한 것인, 채널 코딩 방법.

청구항 4

32개의 행(Row) 및 정보 비트 길이에 해당하는 A개의 열(Column)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩하는 방법에 있어서,

상기 A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 상기 채널 코딩된 결과를 출력 시퀀스로서 출력하는 단계를 포함하며,

상기 A가 10보다 큰 경우, 상기 코드 생성 행렬은,

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$
0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	1
2	1	0	0	1	0	0	1	0	1	1
3	1	0	1	1	0	0	0	0	1	0
4	1	1	1	1	0	0	0	1	0	0
5	1	1	0	0	1	0	1	1	1	0
6	1	0	1	0	1	0	1	0	1	1
7	1	0	0	1	1	0	0	1	1	0
8	1	1	0	1	1	0	0	1	0	1
9	1	0	1	1	1	0	1	0	0	1
10	1	0	1	0	0	1	1	1	0	1
11	1	1	1	0	0	1	1	0	1	0
12	1	0	0	1	0	1	0	1	1	1
13	1	1	0	1	0	1	0	1	0	1
14	1	0	0	0	1	1	0	1	0	0
15	1	1	0	0	1	1	1	1	0	1
16	1	1	1	0	1	1	1	0	0	1
17	1	0	0	1	1	1	0	0	1	0
18	1	1	0	1	1	1	1	1	0	0
19	1	0	0	0	0	1	1	0	0	0
20	1	0	1	0	0	0	1	0	0	0
21	1	1	0	1	0	0	0	0	0	1
22	1	0	0	0	1	0	0	1	1	0
23	1	1	1	0	1	0	0	0	1	1
24	1	1	1	1	1	0	1	1	1	1
25	1	1	0	0	0	1	1	1	0	0
26	1	0	1	1	0	1	0	0	1	1
27	1	1	1	1	0	1	0	1	1	1
28	1	0	1	0	1	1	1	0	1	0
29	1	0	1	1	1	1	1	1	1	0
30	1	1	1	1	1	1	1	1	1	1
31	1	0	0	0	0	0	0	0	0	0

와 같은 제 1 행렬 또는 상기 제 1 행렬의 하나 이상의 행들 사이의 위치가 바뀐 제 2 행렬에,

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0],

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0], 및

[0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1]

와 같은 추가 기본 시퀀스들 중 A-10개의 추가 기본 시퀀스를 열 방향 시퀀스로서 추가한 것인, 채널 코딩 방법.

청구항 5

32개의 행(Row) 및 정보 비트 길이에 해당하는 A개의 열(Column)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩하는 방법에 있어서,

상기 A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 상기 채널 코딩된 결과를 출력 시퀀스로서 출력하는 단

계를 포함하며,

상기 코드 생성 행렬은,

I	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$
0	1	1	0	0	0	0	0	0	0	0	1	1	0
1	1	1	1	0	0	0	0	0	0	1	1	1	0
2	1	0	0	1	0	0	1	0	1	1	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1	1	1
4	1	1	1	1	0	0	0	1	0	0	1	1	1
5	1	1	0	0	1	0	1	1	1	0	1	1	1
6	1	0	1	0	1	0	1	0	1	1	1	1	1
7	1	0	0	1	1	0	0	1	1	0	1	1	1
8	1	1	0	1	1	0	0	1	0	1	1	1	1
9	1	0	1	1	1	0	1	0	0	1	1	1	1
10	1	0	1	0	0	1	1	1	0	1	1	1	1
11	1	1	1	0	0	1	1	0	1	0	1	1	1
12	1	0	0	1	0	1	0	1	1	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1	1	1
14	1	0	0	0	1	1	0	1	0	0	1	0	1
15	1	1	0	0	1	1	1	1	0	1	1	0	1
16	1	1	1	0	1	1	1	0	0	1	0	1	1
17	1	0	0	1	1	1	0	0	1	0	0	1	1
18	1	1	0	1	1	1	1	1	0	0	0	0	0
19	1	0	0	0	0	1	1	0	0	0	0	0	0

와 같은 제 1 행렬 또는 상기 제 1 행렬 중 좌측부터 상기 A길이에 대응하는 기본 시퀀스를 선택한 제 2 행렬에,

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$

0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	1
2	1	0	0	1	0	0	1	0	1	1
3	1	0	1	1	0	0	0	0	1	0
4	1	1	1	1	0	0	0	1	0	0
5	1	1	0	0	1	0	1	1	1	0
6	1	0	1	0	1	0	1	0	1	1
7	1	0	0	1	1	0	0	1	1	0
8	1	1	0	1	1	0	0	1	0	1
9	1	0	1	1	1	0	1	0	0	1
10	1	0	1	0	0	1	1	1	0	1
11	1	1	1	0	0	1	1	0	1	0
12	1	0	0	1	0	1	0	1	1	1
13	1	1	0	1	0	1	0	1	0	1
14	1	0	0	0	1	1	0	1	0	0
15	1	1	0	0	1	1	1	1	0	1
16	1	1	1	0	1	1	1	0	0	1
17	1	0	0	1	1	1	0	0	1	0
18	1	1	0	1	1	1	1	1	0	0
19	1	0	0	0	0	1	1	0	0	0
20	1	0	1	0	0	0	1	0	0	0
21	1	1	0	1	0	0	0	0	0	1
22	1	0	0	0	1	0	0	1	1	0
23	1	1	1	0	1	0	0	0	1	1
24	1	1	1	1	1	0	1	1	1	1
25	1	1	0	0	0	1	1	1	0	0
26	1	0	1	1	0	1	0	0	1	1
27	1	1	1	1	0	1	0	1	1	1
28	1	0	1	0	1	1	1	0	1	0
29	1	0	1	1	1	1	1	1	1	0
30	1	1	1	1	1	1	1	1	1	1
31	1	0	0	0	0	0	0	0	0	0

와 같은 제 3 행렬 또는 상기 제 3 행렬의 하나 이상의 행들 또는 열들 사이의 위치가 바뀐 제 4 행렬 중 상기 제 1 또는 제 2 행렬에 포함되지 않은 잉여 행들에 대응하는 A 비트 길이 시퀀스들을 행 방향 시퀀스로서 추가한 제 5 행렬에 대응하는, 채널 코딩 방법.

청구항 6

제 5 항에 있어서,

상기 잉여 행들에 대응하는 A 비트 길이 시퀀스들은 상기 잉여 행들에 A-10비트 길이의 추가 비트를 추가한 시퀀스들이며,

상기 추가 비트는 상기 추가 비트가 추가되는 기본 시퀀스들이 최소 해밍 거리의 최대값이 10인 조건을 만족하도록 추가되는, 채널 코딩 방법.

청구항 7

제 5 항에 있어서,

상기 추가 비트는 상기 추가 비트가 추가되는 기본 시퀀스들이 0을 10개 포함하도록 추가되는, 채널 코딩 방법.

청구항 8

제 1 항, 제 4 항 또는 제 5 항 중 어느 한 항에 있어서,

상기 A가 14 이하인 경우, 상기 코드 생성 행렬의 기본 시퀀스에는,

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$	$M_{i,13}$
0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	0	0	0	0	1	1	1	0	0
2	1	0	0	1	0	0	1	0	1	1	1	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1	1	1	1
4	1	1	1	1	0	0	0	1	0	0	1	1	1	1
6	1	0	1	0	1	0	1	0	1	1	1	1	1	0
8	1	1	0	1	1	0	0	1	0	1	1	1	1	1
9	1	0	1	1	1	0	1	0	0	1	1	1	1	1
10	1	0	1	0	0	1	1	1	0	1	1	1	1	1
11	1	1	1	0	0	1	1	0	1	0	1	1	1	1
12	1	0	0	1	0	1	0	1	1	1	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1	1	1	1
15	1	1	0	0	1	1	1	1	0	1	1	0	1	1
16	1	1	1	0	1	1	1	0	0	1	0	1	1	1
17	1	0	0	1	1	1	0	0	1	0	0	1	1	1
19	1	0	0	0	0	1	1	0	0	0	0	0	0	1
7	1	0	0	1	1	0	0	1	1	0	1	1	1	1
14	1	0	0	0	1	1	0	1	0	0	1	0	1	1
5	1	1	0	0	1	0	1	1	1	0	1	1	1	0
18	1	1	0	1	1	1	1	1	0	0	0	0	0	1
20	1	0	1	0	0	0	1	0	0	0	1	0	1	1
21	1	1	0	1	0	0	0	0	0	1	1	1	0	0
22	1	0	0	0	1	0	0	1	1	0	1	0	1	1
23	1	1	1	0	1	0	0	0	1	1	1	1	0	1
24	1	1	1	1	1	0	1	1	1	1	0	0	1	0
25	1	1	0	0	0	1	1	1	0	0	1	1	1	0
26	1	0	1	1	0	1	0	0	1	1	0	0	0	1
27	1	1	1	1	0	1	0	1	1	1	0	1	0	0
28	1	0	1	0	1	1	1	0	1	0	0	1	0	1
29	1	0	1	1	1	1	1	1	1	0	0	1	1	0
30	1	1	1	1	1	1	1	1	1	1	1	0	1	0
31	1	0	0	0	0	0	0	0	0	0	0	0	0	1

와 같은 제 6 행렬의 열 방향 시퀀스들 중 좌측으로부터 상기 A 길이에 해당하는 개수의 열 방향 시퀀스가 순차적으로 대응하는, 채널 코딩 방법.

청구항 9

제 1 항, 제 4 항 또는 제 5 항 중 어느 한 항에 있어서,

상기 A가 11 이하인 경우, 상기 코드 생성 행렬의 기본 시퀀스에는,

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$
0	1	1	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	1	1
2	1	0	0	1	0	0	1	0	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1
4	1	1	1	1	0	0	0	1	0	0	1
6	1	0	1	0	1	0	1	0	1	1	1
8	1	1	0	1	1	0	0	1	0	1	1
9	1	0	1	1	1	0	1	0	0	1	1
10	1	0	1	0	0	1	1	1	0	1	1
11	1	1	1	0	0	1	1	0	1	0	1
12	1	0	0	1	0	1	0	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1
15	1	1	0	0	1	1	1	1	0	1	1
16	1	1	1	0	1	1	1	0	0	1	0
17	1	0	0	1	1	1	0	0	1	0	0
19	1	0	0	0	0	1	1	0	0	0	0
7	1	0	0	1	1	0	0	1	1	0	1
14	1	0	0	0	1	1	0	1	0	0	1
5	1	1	0	0	1	0	1	1	1	0	1
18	1	1	0	1	1	1	1	1	0	0	0
20	1	0	1	0	0	0	1	0	0	0	1
21	1	1	0	1	0	0	0	0	0	1	1
22	1	0	0	0	1	0	0	1	1	0	1
23	1	1	1	0	1	0	0	0	1	1	1
24	1	1	1	1	1	0	1	1	1	1	0
25	1	1	0	0	0	1	1	1	0	0	1
26	1	0	1	1	0	1	0	0	1	1	0
27	1	1	1	1	0	1	0	1	1	1	0
28	1	0	1	0	1	1	1	0	1	0	0
29	1	0	1	1	1	1	1	1	1	0	0
30	1	1	1	1	1	1	1	1	1	1	1
31	1	0	0	0	0	0	0	0	0	0	0

와 같은 제 7 행렬의 열 방향 시퀀스들 중 좌측으로부터 상기 A 길이에 해당하는 개수의 열 방향 시퀀스가 순차적으로 대응하는, 채널 코딩 방법.

청구항 10

제 1 항, 제 4 항 또는 제 5 항 중 어느 한 항에 있어서,

상기 출력 시퀀스의 비트 수가 32비트보다 큰 경우,

상기 코드 생성 행렬의 각 기본 시퀀스를 소정 횟수 반복하고, 반복된 각 기본 시퀀스 중 상기 출력 시퀀스의 비트 수에 대응하는 길이만큼을 이용하여 채널 코딩을 수행하는, 채널 코딩 방법.

청구항 11

제 1 항, 제 4 항 또는 제 5 항 중 어느 한 항에 있어서,

상기 출력 시퀀스의 비트 수가 32비트보다 큰 경우,

상기 출력 시퀀스는 상기 채널 코딩된 결과를 순환적으로 반복하여 획득되는, 채널 코딩 방법.

청구항 12

제 1 항, 제 4 항 또는 제 5 항 중 어느 한 항에 있어서,

상기 정보 비트는 채널 품질 정보(CQI) 또는 프리코딩 행렬 인덱스(PMI) 중 어느 하나 이상에 해당하는 비트인, 채널 코딩 방법.

청구항 13

제 1 항, 제 4 항 또는 제 5 항 중 어느 한 항에 있어서,

상기 출력 시퀀스는 물리 상향링크 공유 채널(PUSCH)를 통해 전송되는, 채널 코딩 방법.

청구항 14

32개의 행(row) 및 정보 비트 길이에 해당하는 A개의 열(column)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩하는 방법에 있어서,

상기 A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 상기 채널 코딩된 결과를 출력 시퀀스로서 출력하는 단계를 포함하며,

상기 코드 생성 행렬의 기본 시퀀스에는,

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$
0	1	1	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	1	1
2	1	0	0	1	0	0	1	0	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1
4	1	1	1	1	0	0	0	1	0	0	1
5	1	1	0	0	1	0	1	1	1	0	1
6	1	0	1	0	1	0	1	0	1	1	1
7	1	0	0	1	1	0	0	1	1	0	1
8	1	1	0	1	1	0	0	1	0	1	1
9	1	0	1	1	1	0	1	0	0	1	1
10	1	0	1	0	0	1	1	1	0	1	1
11	1	1	1	0	0	1	1	0	1	0	1
12	1	0	0	1	0	1	0	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1
14	1	0	0	0	1	1	0	1	0	0	1
15	1	1	0	0	1	1	1	1	0	1	1
16	1	1	1	0	1	1	1	0	0	1	0
17	1	0	0	1	1	1	0	0	1	0	0
18	1	1	0	1	1	1	1	1	0	0	0
19	1	0	0	0	0	1	1	0	0	0	0
20	1	0	1	0	0	0	1	0	0	0	1
21	1	1	0	1	0	0	0	0	0	1	1
22	1	0	0	0	1	0	0	1	1	0	1
23	1	1	1	0	1	0	0	0	1	1	1
24	1	1	1	1	1	0	1	1	1	1	0
25	1	1	0	0	0	1	1	1	0	0	1
26	1	0	1	1	0	1	0	0	1	1	0
27	1	1	1	1	0	1	0	1	1	1	0
28	1	0	1	0	1	1	1	0	1	0	0
29	1	0	1	1	1	1	1	1	1	0	0
30	1	1	1	1	1	1	1	1	1	1	1
31	1	0	0	0	0	0	0	0	0	0	0

와 같은 행렬의 열 방향 시퀀스들 중 좌측으로부터 상기 A 길이에 해당하는 개수의 열 방향 시퀀스가 순차적으로 대응하는, 채널 코딩 방법.

명 세 서

발명의 상세한 설명

기 술 분 야

<1> 이하의 설명은 이동통신 시스템의 부호화 방법에 대한 것으로서, 구체적으로 블록 코드(Block Code)를 이용하여 다양한 길이를 가지는 정보에 효율적으로 채널 코딩을 수행하는 방법에 대한 것이다.

배 경 기 술

<2> 이하에서는 먼저 기본적인 부호화(또는 코딩) 이론들 중에서 본 발명의 설명을 위해 필요한 내용에 관해서 살펴보도록 한다.

<3> 일반적인 이진 오류정정코드(error correction code)를 [n, k, d] 라고 표시하면, n은 부호화된 부호어(符號語, codeword)의 비트 수, k는 부호화 이전의 정보 비트수, d는 부호어들 간의 거리 중 최소값을 의미한다. 여기서

고려하는 것은 이진 부호이므로, 부호어의 부호길이는 2^n이고, 부호화된 부호어의 총 개수는 2^k가 된다. 또한, 편의상 최소거리가 내용상 크게 중요하지 않은 경우는 [n, k] 부호라고 표시하기도 한다. 본 설명에서는 특별한 언급이 없는 한 n, k 그리고 d가 나타내는 값은 상술한 내용으로 고정하기로 한다.

<4> 이때, 이하의 설명에 있어서 X개의 행 및 Y개의 열로 구성되는 행렬 형태의 블록 코드를 (X, Y)로 표시하는 경우와 혼동하지 말아야 한다.

<5> 한편 부호율 R은 정보비트 수를 부호어의 비트 수로 나눈 값으로 정의된다. 즉, R=k/n으로 정의된다.

<6> 이하 해밍 거리(Hamming distance)에 대해 설명한다.

<7> 해밍 거리는 같은 비트 수를 갖는 2진 부호 사이에 대응되는 비트 값이 일치하지 않는 것의 개수를 의미한다. 일반적으로 해밍 거리 d가 d=2a+1이면 a개의 오류를 정정할 수 있다. 예를 들어 두 개의 부호어가 101011와 110010이라면, 두 부호어의 해밍 거리는 3이다.

<8> 한편, 부호화 이론에 있어서 최소거리(minimum distance)는 부호에 속하는 임의의 두 부호어 사이의 거리의 최소값을 의미한다. 이와 같은 최소 거리는 부호의 좋은 정도를 가리키는 중요한 평가량의 하나이며, 거리로는 상술한 해밍 거리가 쓰일 때가 많다. 부호화 과정을 거쳐 생성된 부호어들 간의 거리가 멀수록 해당 부호어가 다른 부호어로 판단될 확률이 낮아지므로, 부호화 성능이 좋아지게 된다. 또한, 전체 부호의 성능은 가장 나쁜 성능을 갖는 부호어들간의 거리 즉, 부호어들 중에서 최소 거리에 의해 부호의 성능이 평가된다. 결국, 정리하면 좋은 부호는 최소 거리가 최대화된 부호가 좋은 성능을 보이게 된다.

<9> 차세대 이동통신 시스템에서 제어 정보는 시스템의 구성 및 전송 채널의 정보 등을 전송함으로써 시스템의 성능을 결정하는 매우 중요한 정보이다. 이러한 제어 정보는 시스템의 자원을 되도록 적게 사용하기 위해 짧은 길이로 구성되며, 채널 오류에 강인한 우수한 부호화 기법을 이용하여 부호화된 후 전송된다. 일례로, 3GPP 이동통신 시스템에서 제어 정보를 위한 부호화 기법으로는 RM (Reed-Muller) 부호에 기반한 짧은 길이의 블록 부호, 테일 바이팅 컨벌루셔널(tail-biting convolutional) 부호, 심플렉스(simplex) 부호의 반복 부호 등이 고려되고 있다.

<10> 한편 상술한 3GPP 이동통신 시스템의 진보 형태인 3GPP LTE 시스템에서 제어 정보는 블록 코드(Block Code)를 이용한 부호화를 거쳐 전송되는 것이 논의되고 있다. 구체적으로 전송되는 정보 비트의 길이가 A라 할 경우, 특정 채널(예를 들어, 물리 상향링크 제어 채널(이하 "PUCCH")의 전송에 있어서 20개의 행과 A개의 열로 구성되는 블록 코드(또는 (20, A) 블록 코드)를 이용하여 채널 코딩을 수행한 후 전송되는 것이 논의되고 있다. 또한, 3GPP LTE 시스템에 있어서 상향링크 제어 정보는 상기 PUCCH뿐만 아니라 상향링크 공유 채널(이하 "PUSCH")를 통해서도 전송되고 있으며, 이와 같이 PUSCH를 통해 전송되는 제어 정보는 32개의 행과 A개의 열로 구성되는 블록 코드(또는 (32, A) 블록 코드)를 이용하여 채널 코딩을 수행한 후 전송되는 것이 논의되고 있다.

<11> 한편, (32, A)의 형태를 가지는 블록 코드에는 수많은 형태가 가능하며, 각각의 블록 코드 모두에 대해 다양한 길이를 가지는 정보 비트의 부호화 성능을 각각 확인하여 최적의 형태를 찾는 것은 쉬운 일이 아니다.

발명의 내용

해결 하고자하는 과제

<12> 상술한 바와 같은 문제를 해결하기 위해 본 발명의 일 실시형태에서는 다양한 길이를 갖는 정보의 효율적인 (32, A) 블록 부호화 방법을 제안한다. 즉, 정보 비트의 길이가 다양하게 변하고, 부호화된 부호어의 비트 길이도 다양하게 변하는 상황에서, 상술한 바와 같이 다양한 비트 길의 조합을 효과적으로 지원하는 (32, A) 블록 부호화 방법을 제안하고자 한다.

<13> 한편, 실제 사용되는 부호화 비트 수는 32 이하일 수 있으며, 정보 비트수도 다양하게 변경될 수 있다. 따라서, 본 발명에 따른 실시형태들에서는 특정 길이의 정보 비트/부호화 비트 수에 대해 제안된 블록 부호에서 필요한 만큼 일부분만을 효율적으로 사용하는 방법 및 이와 반대로 상기 특정 길이보다 긴 길이의 부호화가 필요한 경우 상기 특정 길이에 기반한 블록 부호의 반복을 통해 긴 길이의 부호화를 수행하는 방법을 제안한다.

과제 해결수단

<14> 상술한 바와 같은 과제를 해결하기 위해 본 발명의 일 실시형태에서는 32개의 행(Row) 및 정보 비트 길이에 해

당하는 A개의 열(Column)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩하는 방법으로서, 상기 A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 상기 채널 코딩된 결과를 출력 시퀀스로서 출력하는 단계를 포함하며, 상기 A가 10보다 큰 경우, 상기 코드 생성 행렬은 TFCI(Transport Format Combination Indicator) 정보의 코딩에 사용된 32개의 행 및 10개의 열로 구성된 TFCI 코드 생성 행렬에 대응하는 제 1 행렬 또는 상기 제 1 행렬의 하나 이상의 행들 사이의 위치가 바뀐 제 2 행렬에 최소 해밍 거리의 최대값이 10인 조건을 만족하는 추가 기본 시퀀스들 중 A-10개의 추가 기본 시퀀스를 열 방향 시퀀스로서 추가한 것인 것을 특징으로 하는 채널 코딩 방법을 제안한다.

<15> 이때, 상기 추가 기본 시퀀스는 0을 10개 포함하는 기본 시퀀스일 수 있다.

<16> 또한, 상기 제 2 행렬은, 상기 제 2 행렬의 하단 12행을 제외한 제 3 행렬이 물리상향링크 제어 패널(PUCCH) 전송을 위한 코드 생성 행렬에 대응하도록 상기 제 1 행렬의 하나 이상의 행들 사이의 위치를 변경한 것일 수 있다.

<17> 한편, 상술한 바와 같은 과제를 해결하기 위한 본 발명의 다른 일 실시형태에서는 32개의 행(Row) 및 정보 비트 길이에 해당하는 A개의 열(Column)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩하는 방법으로서, 상기 A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 상기 채널 코딩된 결과를 출력 시퀀스로서 출력하는 단계를 포함하며, 상기 A가 10보다 큰 경우, 상기 코드 생성 행렬은,

<18>

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$
0	1	1	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	1	1
2	1	0	0	1	0	0	1	0	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1
4	1	1	1	1	0	0	0	1	0	0	1
5	1	1	0	0	1	0	1	1	1	0	1
6	1	0	1	0	1	0	1	0	1	1	1

7	1	0	0	1	1	0	0	1	1	0	1
8	1	1	0	1	1	0	0	1	0	1	1
9	1	0	1	1	1	0	1	0	0	1	1
10	1	0	1	0	0	1	1	1	0	1	1
11	1	1	1	0	0	1	1	0	1	0	1
12	1	0	0	1	0	1	0	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1
14	1	0	0	0	1	1	0	1	0	0	1
15	1	1	0	0	1	1	1	1	0	1	1
16	1	1	1	0	1	1	1	0	0	1	0
17	1	0	0	1	1	1	0	0	1	0	0
18	1	1	0	1	1	1	1	1	0	0	0
19	1	0	0	0	0	1	1	0	0	0	0
20	1	0	1	0	0	0	1	0	0	0	1
21	1	1	0	1	0	0	0	0	0	1	1
22	1	0	0	0	1	0	0	1	1	0	1
23	1	1	1	0	1	0	0	0	1	1	1
24	1	1	1	1	1	0	1	1	1	1	0
25	1	1	0	0	0	1	1	1	0	0	1
26	1	0	1	1	0	1	0	0	1	1	0
27	1	1	1	1	0	1	0	1	1	1	0
28	1	0	1	0	1	1	1	0	1	0	0
29	1	0	1	1	1	1	1	1	1	0	0
30	1	1	1	1	1	1	1	1	1	1	1
31	1	0	0	0	0	0	0	0	0	0	0

<19>

<20> 와 같은 제 1 행렬 또는 상기 제 1 행렬의 하나 이상의 행들 사이의 위치가 바뀐 제 2 행렬에,

<21> [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0],

<22> [0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0], 및

<23> [0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1]

<24> 와 같은 추가 기본 시퀀스들 중 A-10개의 추가 기본 시퀀스를 열 방향 시퀀스로서 추가한 것임을 특징으로 하는 채널 코딩 방법을 제안한다.

<25> 한편, 상술한 바와 같은 과제를 해결하기 위한 본 발명의 또 다른 일 실시형태에서는 32개의 행(Row) 및 정보 비트 길이에 해당하는 A개의 열(Row)을 포함하는 코드 생성 행렬을 이용하여 상기 정보 비트를 채널 코딩하는 방법으로서, 상기 A 길이를 가지는 정보 비트를 상기 코드 생성 행렬의 각 열에 대응하는 32 비트 길이를 가지는 기본 시퀀스(Basis Sequence)를 이용하여 채널 코딩을 수행하여, 상기 채널 코딩된 결과를 출력 시퀀스로서 출력하는 단계를 포함하며, 상기 코드 생성 행렬은,

I	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$
0	1	1	0	0	0	0	0	0	0	0	1	1	0
1	1	1	1	0	0	0	0	0	0	1	1	1	0
2	1	0	0	1	0	0	1	0	1	1	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1	1	1
4	1	1	1	1	0	0	0	1	0	0	1	1	1
5	1	1	0	0	1	0	1	1	1	0	1	1	1
6	1	0	1	0	1	0	1	0	1	1	1	1	1
7	1	0	0	1	1	0	0	1	1	0	1	1	1
8	1	1	0	1	1	0	0	1	0	1	1	1	1
9	1	0	1	1	1	0	1	0	0	1	1	1	1
10	1	0	1	0	0	1	1	1	0	1	1	1	1
11	1	1	1	0	0	1	1	0	1	0	1	1	1
12	1	0	0	1	0	1	0	1	1	1	1	1	1

<26>

13	1	1	0	1	0	1	0	1	0	1	1	1	1
14	1	0	0	0	1	1	0	1	0	0	1	0	1
15	1	1	0	0	1	1	1	1	0	1	1	0	1
16	1	1	1	0	1	1	1	0	0	1	0	1	1
17	1	0	0	1	1	1	0	0	1	0	0	1	1
18	1	1	0	1	1	1	1	1	0	0	0	0	0
19	1	0	0	0	0	1	1	0	0	0	0	0	0

<27>

<28> 와 같은 제 1 행렬 또는 상기 제 1 행렬 중 좌측부터 상기 A길이에 대응하는 기본 시퀀스를 선택한 제 2 행렬에,

<29>

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$
0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	1
2	1	0	0	1	0	0	1	0	1	1
3	1	0	1	1	0	0	0	0	1	0
4	1	1	1	1	0	0	0	1	0	0
5	1	1	0	0	1	0	1	1	1	0
6	1	0	1	0	1	0	1	0	1	1
7	1	0	0	1	1	0	0	1	1	0
8	1	1	0	1	1	0	0	1	0	1
9	1	0	1	1	1	0	1	0	0	1
10	1	0	1	0	0	1	1	1	0	1
11	1	1	1	0	0	1	1	0	1	0
12	1	0	0	1	0	1	0	1	1	1
13	1	1	0	1	0	1	0	1	0	1
14	1	0	0	0	1	1	0	1	0	0
15	1	1	0	0	1	1	1	1	0	1
16	1	1	1	0	1	1	1	0	0	1
17	1	0	0	1	1	1	0	0	1	0
18	1	1	0	1	1	1	1	1	0	0

<30>

19	1	0	0	0	0	1	1	0	0	0
20	1	0	1	0	0	0	1	0	0	0
21	1	1	0	1	0	0	0	0	0	1
22	1	0	0	0	1	0	0	1	1	0
23	1	1	1	0	1	0	0	0	1	1
24	1	1	1	1	1	0	1	1	1	1
25	1	1	0	0	0	1	1	1	0	0
26	1	0	1	1	0	1	0	0	1	1
27	1	1	1	1	0	1	0	1	1	1
28	1	0	1	0	1	1	1	0	1	0
29	1	0	1	1	1	1	1	1	1	0
30	1	1	1	1	1	1	1	1	1	1
31	1	0	0	0	0	0	0	0	0	0

<31> 와 같은 제 3 행렬 또는 상기 제 3 행렬의 하나 이상의 행들 또는 열들 사이의 위치가 바뀐 제 4 행렬 중 상기 제 1 또는 제 2 행렬에 포함되지 않은 잉여 행들에 대응하는 A 비트 길이 시퀀스들을 행 방향 시퀀스로서 추가한 제 5 행렬에 대응하는 것을 특징으로 하는 채널 코딩 방법을 제안한다.

<32> 이때, 상기 잉여 행들에 대응하는 A 비트 길이 시퀀스들은 상기 잉여 행들에 A-10비트 길이의 추가 비트를 추가한 시퀀스들이며, 상기 추가 비트는 상기 추가 비트가 추가되는 기본 시퀀스들이 최소 해밍 거리의 최대값이 10인 조건을 만족하도록 추가되는 것이 바람직하다. 구체적으로 이와 같은 추가 비트는 상기 추가 비트가 추가되는 기본 시퀀스들이 0을 10개 포함하도록 추가되는 것일 수 있다.

<33> 한편, 상기 A가 14 이하인 경우, 상기 코드 생성 행렬의 기본 시퀀스에는,

<34>

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,1}$	$M_{i,1}$	$M_{i,1}$	$M_{i,1}$

											0	1	2	3
0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	0	0	0	0	1	1	1	0	0
2	1	0	0	1	0	0	1	0	1	1	1	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1	1	1	1
4	1	1	1	1	0	0	0	1	0	0	1	1	1	1
6	1	0	1	0	1	0	1	0	1	1	1	1	1	0
8	1	1	0	1	1	0	0	1	0	1	1	1	1	1
9	1	0	1	1	1	0	1	0	0	1	1	1	1	1
10	1	0	1	0	0	1	1	1	0	1	1	1	1	1
11	1	1	1	0	0	1	1	0	1	0	1	1	1	1
12	1	0	0	1	0	1	0	1	1	1	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1	1	1	1
15	1	1	0	0	1	1	1	1	0	1	1	0	1	1
16	1	1	1	0	1	1	1	0	0	1	0	1	1	1
17	1	0	0	1	1	1	0	0	1	0	0	1	1	1
19	1	0	0	0	0	1	1	0	0	0	0	0	0	1
7	1	0	0	1	1	0	0	1	1	0	1	1	1	1
14	1	0	0	0	1	1	0	1	0	0	1	0	1	1
5	1	1	0	0	1	0	1	1	1	0	1	1	1	0
18	1	1	0	1	1	1	1	1	0	0	0	0	0	1
20	1	0	1	0	0	0	1	0	0	0	1	0	1	1
21	1	1	0	1	0	0	0	0	0	1	1	1	0	0
22	1	0	0	0	1	0	0	1	1	0	1	0	1	1
23	1	1	1	0	1	0	0	0	1	1	1	1	0	1
24	1	1	1	1	1	0	1	1	1	1	0	0	1	0
25	1	1	0	0	0	1	1	1	0	0	1	1	1	0
26	1	0	1	1	0	1	0	0	1	1	0	0	0	1
27	1	1	1	1	0	1	0	1	1	1	0	1	0	0
28	1	0	1	0	1	1	1	0	1	0	0	1	0	1
29	1	0	1	1	1	1	1	1	1	0	0	1	1	0
30	1	1	1	1	1	1	1	1	1	1	1	0	1	0
31	1	0	0	0	0	0	0	0	0	0	0	0	0	1

<35>

<36> 와 같은 제 6 행렬의 열 방향 시퀀스들 중 좌측으로부터 상기 A 길이에 해당하는 개수의 열 방향 시퀀스가 순차적으로 대응하는 것일 수 있다.

<37> 또한, 만일 상기 A가 11 이하인 경우, 상기 코드 생성 행렬의 기본 시퀀스에는,

I	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$
0	1	1	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	1	1
2	1	0	0	1	0	0	1	0	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1
4	1	1	1	1	0	0	0	1	0	0	1
6	1	0	1	0	1	0	1	0	1	1	1
8	1	1	0	1	1	0	0	1	0	1	1
9	1	0	1	1	1	0	1	0	0	1	1
10	1	0	1	0	0	1	1	1	0	1	1
11	1	1	1	0	0	1	1	0	1	0	1
12	1	0	0	1	0	1	0	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1
15	1	1	0	0	1	1	1	1	0	1	1
16	1	1	1	0	1	1	1	0	0	1	0
17	1	0	0	1	1	1	0	0	1	0	0
19	1	0	0	0	0	1	1	0	0	0	0
7	1	0	0	1	1	0	0	1	1	0	1
14	1	0	0	0	1	1	0	1	0	0	1
5	1	1	0	0	1	0	1	1	1	0	1
18	1	1	0	1	1	1	1	1	0	0	0
20	1	0	1	0	0	0	1	0	0	0	1
21	1	1	0	1	0	0	0	0	0	1	1
22	1	0	0	0	1	0	0	1	1	0	1
23	1	1	1	0	1	0	0	0	1	1	1
24	1	1	1	1	1	0	1	1	1	1	0

<38>

25	1	1	0	0	0	1	1	1	0	0	1
26	1	0	1	1	0	1	0	0	1	1	0
27	1	1	1	1	0	1	0	1	1	1	0
28	1	0	1	0	1	1	1	0	1	0	0
29	1	0	1	1	1	1	1	1	1	0	0
30	1	1	1	1	1	1	1	1	1	1	1
31	1	0	0	0	0	0	0	0	0	0	0

<39>

<40> 와 같은 제 7 행렬의 열 방향 시퀀스들 중 좌측으로부터 상기 A 길이에 해당하는 개수의 열 방향 시퀀스가 순차적으로 대응할 수 있다.

<41> 아울러, 상기 출력 시퀀스의 비트 수가 32비트보다 큰 경우, 상기 코드 생성 행렬의 각 기본 시퀀스를 소정 횟수 반복하고, 반복된 각 기본 시퀀스 중 상기 출력 시퀀스의 비트 수에 대응하는 길이만큼을 이용하여 채널 코딩을 수행하는 것일 수 있다. 즉, 상기 출력 시퀀스의 비트 수가 32비트보다 큰 경우, 상기 출력 시퀀스는 상기 채널 코딩된 결과를 순환적으로 반복하여 획득되는 것일 수 있다.

<42> 상술한 실시형태들에 있어서 상기 정보 비트는 채널 품질 정보(CQI) 또는 프리코딩 행렬 인덱스(PMI) 중 어느

하나 이상에 해당하는 비트일 수 있으며, 상기 출력 시퀀스는 물리 상향링크 공유 채널(PUSCH)를 통해 전송될 수 있다.

효 과

<43> 상술한 바와 같은 본 발명의 각 실시형태에 따르면, 기존에 3GPP 시스템에서 TFCI 정보 코딩에 사용되었던 코드 생성 행렬 및/또는 PUCCH 전송을 위한 (20, A) 코드 생성 행렬을 재사용하여, 손쉽게 (32, k) 블록 코딩을 구현할 수 있으며, 이에 의해 생성되는 부호어들간에 최대최소거리를 증가시켜 성능을 향상시킬 수 있다.

발명의 실시를 위한 구체적인 내용

<44> 이하, 본 발명에 따른 바람직한 실시 형태를 첨부된 도면을 참조하여 상세하게 설명한다. 첨부된 도면과 함께 이하에 개시될 상세한 설명은 본 발명의 예시적인 실시형태를 설명하고자 하는 것이며, 본 발명이 실시될 수 있는 유일한 실시형태를 나타내고자 하는 것이 아니다. 예를 들어, 이하의 설명은 이해를 돕기 위해 상술한 3GPP LTE (3rd Generation Partnership Project Long Term Evolution) 시스템에 적용되는 구체적인 예를 들어 설명하나, 본 발명은 3GPP LTE 시스템뿐만 아니라 일반적으로 다양한 길이를 가지는 제어 정보를 블록 코드를 이용하여 채널 코딩을 수행할 필요가 있는 임의의 통신 시스템에 적용될 수 있다.

<45> 이하의 상세한 설명은 본 발명의 완전한 이해를 제공하기 위해서 구체적 세부사항을 포함한다. 그러나, 당업자는 본 발명이 이러한 구체적 세부사항 없이도 실시될 수 있음을 안다. 몇몇 경우, 본 발명의 개념이 모호해지는 것을 피하기 위하여 공지의 구조 및 장치는 생략되거나, 각 구조 및 장치의 핵심기능을 중심으로 한 블록도 형식으로 도시된다. 또한, 본 명세서 전체에서 동일한 구성요소에 대해서는 동일한 도면 부호를 사용하여 설명한다.

<46> 우선, 본 발명에서 공통으로 고려해야 할 점에 대해서 간단히 언급한다.

<47> [n, k] 부호는 상술한 바와 같이 부호화 비트수가 n 이고 정보 비트수가 k 인 부호를 의미한다.

<48> 여기서는 앞으로 특별한 언급이 없는 한, 편의상 부호의 생성 행렬을 기본 시퀀스(basis sequence)형태의 표로 표시하기로 한다. 실제 부호화 방법은 3GPP release 99 의 TFCI 부호에서와 비슷한 방법을 사용한다. 즉, 정보 비트를 왼쪽 기본 시퀀스 에 차례대로 할당하고, 기본 시퀀스와 정보 비트를 곱한 결과에 대응하는 시퀀스를 정보 비트 수만큼 2 진 연산으로 더하여(Exclusive OR sum) 부호화 비트를 생성하는 방법을 이용한다.

<49> 상술한 방법으로 부호를 나타내면 정보비트의 수가 가변적일 때에도 행렬 형태의 하나의 기본 시퀀스 표를 사용하여 부호화가 가능한 장점이 있다. 이러한 장점을 사용하여 다양한 정보 비트수의 지원이 가능하다. 따라서, 기본 시퀀스 표 또는 코드 생성 행렬은 최대크기의 정보 비트 수를 고려하여 표시되었다. 만일, 실제 응용에서 필요한 최대 정보비트수가 이하에서 제시한 크기보다 작다면, 해당 최대 비트 수 이상을 위한 기본 시퀀스를 삭제한 표를 사용하는 것이 바람직하다.

<50> 한편, 부호화 이론에 의해서 생성부호에서 0과 1을 바꾸는 것은 부호의 특성에 영향을 끼치지 않는다. 따라서, 기본 시퀀스 표에서 0과 1을 서로 바꾼 표도 동일한 부호이다. 또한, 부호화 이론에 의해서 부호화된 비트의 순서를 바꾸는 것도 동일한 부호 특성을 나타낸다. 따라서, 기본 시퀀스 표에서 행의 위치를 서로 바꾼 것도 동일한 부호이다.

<51> 본 발명에 따른 이하의 실시형태들에서 제시한 기본 시퀀스는 정보비트의 수를 가변으로 하도록 설계되었을 뿐만 아니라, 부호화 비트수도 가변으로 할 수 있도록 설계하였다. 따라서 특정 기본 시퀀스 표에서 특정 열을 삭제한 부호도 본 발명의 착상시 이미 고려된 부호이다. 즉, 간단한 일례로, 기본 시퀀스 표가 (32,14)와 같은 행태를 가진다면, 아래부터 연속적인 12개의 행을 삭제하고, 오른쪽부터 3개의 열을 삭제한 (20, 11) 형태의 기본 시퀀스 표 역시 상기 (32,14) 기본 시퀀스 표의 한 가지 적용 예이다. 정리하면, 본 발명에서는 기본 시퀀스 표의 행과 열이 가장 큰 크기를 기준으로 이루어졌으며, 작은 크기의 행과 열은 상기 큰 크기의 기본 시퀀스 표의 행과 열을 오른쪽 또는 하단부터 차례대로 삭제하여 사용하도록 한다. 물론 앞서 설명한 바와 같이, 줄어든 크기의 기본 시퀀스 표에서 행과 열의 위치를 바꾸거나, 0과 1을 교환한 표도 동일한 부호임에 유의해야 한다.

<52> 참고로, 본 발명의 설명에 있어서의 표기는, 상기 기본 시퀀스 표에서 정보 비트는 왼쪽 열부터 시작하여 오른쪽 열로 순차적으로 대응되고, 부호화 비트는 가장 상단의 행부터 시작하여 가장 하단의 행 방향으로 순차적으로 대응되는 것으로 한다. 또한, 상술한 "기본 시퀀스 표" 는 "코드 생성 행렬" 등 다른 용어로 지칭될 수도 있다.

<53> 한편, 특정 채널 추정 방법 등에서 특정한 패턴의 기본 시퀀스는 사용하지 않는 것이 바람직할수도 있다. 이러한 경우에는 본 발명에 따른 이하의 실시형태들에서 제안한 표들 중에서, 시스템에 따라서 특정 기본 시퀀스를 제거한 표가 이용되는 것을 고려할 수 있다. 이러한 경우는 부호화 관점에서 해당 기본 시퀀스는 언제나 0인 경우로 고려 가능하므로, 부호성능은 동일하고, 정보비트수만 줄어든 형태이다.

<54> 따라서, 본 발명에 따른 이하의 실시형태들에서 제안한 기본 시퀀스 표에서 특정 기본 시퀀스를 제거한 기본 시퀀스 표 또는 코드 생성 행렬도 본 발명의 착상시 이미 고려된 것이다.

<55> 상술한 바와 같은 본 발명의 일 측면에서는 (32, A) 형태의 블록 코딩 방식을 제안한다. 구체적으로 이하에서는 정보 비트의 길이가 최대 14비트인 경우를 고려하여 (32, 14) 형태의 블록 코딩을 고려한다. 다만, 정보 비트의 길이에 따라 (32, 14) 형태의 코드 생성 행렬 중 일부 기본 시퀀스들만 이용할 수 있음은 상술한 바와 같다.

<56> 이를 위해 다양한 방법이 존재할 수 있으나, 이하에서 설명할 실시형태들에서는 최대한 기존에 제안된 블록 코드들을 이용하여 기존 부호들과 최대한 공통점을 유지하도록 코드를 설계하는 것을 제안한다. 구체적으로, 기존에 제안된 코드로는 3GPP release 99에서 이용되는 TFCI 정보 코딩을 위한 (32, 10) 형태의 블록 코드 및 본 발명자에 의해 본원의 우선권 주장의 기초가 되는 미국 가출원 제 61/016,492 호(GENERATION METHOD OF VARIOUS SHORT LENGTH BLOCK CODES WITH NESTED STRUCTURE BY PUNCTURING A BASE CODE) 등에 제안된 PUCCH 전송용 (20, 14) 블록 코드를 이용하는 것을 제안하여, 구체적인 내용은 이하에서 후술한다.

<57> 먼저, 상술한 (32, 10) 형태의 블록 코드 및 PUCCH 전송용 (20, 14) 블록 코드에 대해 설명한다.

<58> **(32, 10) TFCI 블록 코드 및 (20, 14) 블록 코드**

<59> 본 실시형태에서 이용되는 (20, 14) 블록 코드는 먼저 (32, 10) 구조를 가지는 TFCI 코드 생성 행렬에서 (20, 10) 블록 코드를 생성한 후, 이 블록 코드에 4개의 기본 시퀀스를 추가하여 (20, 14) 블록 코드를 생성하는 것을 제안한다.

<60> 먼저, (20, 10) 블록 코드는 3GPP Rel' 99에서 TFCI(Transport Format Combination Indicator) 정보의 채널코딩에 사용되었던 (32,10) 부호의 생성행렬을 기반으로 하여, 부호화되는 부호어의 길이에 따라 천공(Puncturing)이 된 형태를 가지도록 설계하는 것을 제안한다.

<61> 이러한 (32,10) TFCI 정보 코드의 재사용은 여러 가지 장점을 갖는다. 첫째로, TFCI 정보 코드 자체가 Reed-Muller 부호를 기초로 하여 설계 되었으므로, 천공된 TFCI 부호도 변형된 Reed-Muller부호 구조를 갖는다. 이러한 Reed-Muller 기반의 부호는 복호 과정에서 빠른 하다마드 변환(Fast Hadamard Transform)과 같은 방법을 사용하여 빠른 복호가 가능한 장점이 있다. 두 번째로, TFCI 부호화 방식은 다양한 길이의 정보비트와 부호화 비트를 지원한다. 이렇게 정보 비트의 길이나 부호화 비트의 길이가 다양하게 변화 가능하므로, 현재 3GPP LTE의 CQI 전송을 위한 요구사항을 잘 만족하게 된다.

<62> 아래 표 1은 3GPP Rel' 99에서 TFCI 정보 채널코딩에 사용되었던 길이가 32 비트이고 $d_{\min}$ = 12 인 부호어를 생성하는 (32,10) 부호를 생성행렬을 나타낸다.

<63> [표 1]

TFCI Index	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$
0	1	0	0	0	0	1	0	0	0	0
1	0	1	0	0	0	1	1	0	0	0
2	1	1	0	0	0	1	0	0	0	1
3	0	0	1	0	0	1	1	0	1	1
4	1	0	1	0	0	1	0	0	0	1
5	0	1	1	0	0	1	0	0	1	0
6	1	1	1	0	0	1	0	1	0	0
7	0	0	0	1	0	1	0	1	1	0
8	1	0	0	1	0	1	1	1	1	0
9	0	1	0	1	0	1	1	0	1	1
10	1	1	0	1	0	1	0	0	1	1
11	0	0	1	1	0	1	0	1	1	0
12	1	0	1	1	0	1	0	1	0	1
13	0	1	1	1	0	1	1	0	0	1
14	1	1	1	1	0	1	1	1	1	1
15	1	0	0	0	1	1	1	1	0	0
16	0	1	0	0	1	1	1	1	0	1

<64>

17	1	1	0	0	1	1	1	0	1	0
18	0	0	1	0	1	1	0	1	1	1
19	1	0	1	0	1	1	0	1	0	1
20	0	1	1	0	1	1	0	0	1	1
21	1	1	1	0	1	1	0	1	1	1
22	0	0	0	1	1	1	0	1	0	0
23	1	0	0	1	1	1	1	1	0	1
24	0	1	0	1	1	1	1	0	1	0
25	1	1	0	1	1	1	1	0	0	1
26	0	0	1	1	1	1	0	0	1	0
27	1	0	1	1	1	1	1	1	0	0
28	0	1	1	1	1	1	1	1	1	0
29	1	1	1	1	1	1	1	1	1	1
30	0	0	0	0	0	1	0	0	0	0
31	0	0	0	0	1	1	1	0	0	0

<65>

<66> 일반적으로 블록 코드에 있어서 행 서로간 또는 열 서로간에 위치를 교환하더라도 생성되는 부호어에 성능 차이가 없는 것으로 알려져 있다. 이와 같은 점을 이용하여 이하 표 2는 상술한 TFCI 정보 코딩에 이용된 (32, 10) 블록 코드와 등가인 블록 코드를 나타낸다.

<67> [표 2]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$g_{10by32}[1]$	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
$g_{10by32}[2]$	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
$g_{10by32}[3]$	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
$g_{10by32}[4]$	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$g_{10by32}[5]$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$g_{10by32}[6]$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$g_{10by32}[7]$	1	1	1	0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	1	0	0
$g_{10by32}[8]$	1	1	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0
$g_{10by32}[9]$	1	1	0	1	0	1	0	0	1	1	0	1	1	0	0	0	1	0	0	1	1	1	1	1	0	1	0	1	0	0	0	0
$g_{10by32}[10]$	1	0	0	0	1	0	1	0	1	1	1	1	0	1	0	0	1	1	1	0	1	1	0	0	0	0	1	1	1	0	0	0

<68>

<69> 상기 표 2에 표시된 블록 코드는 상기 TFCI 코딩에 이용된 (32, 10) 코드의 행과 열의 위치가 변경되었으며, 몇몇 개의 열(표 1의 TFCI 정보 코드를 기준으로는 행)간에 서로 위치가 바뀌어 있는 형태를 나타내고 있다.

<70> 즉, 본 실시형태에서는 상술한 바와 같이 (32, 10)의 형태를 가지는 TFCI 정보 코드(표 1) 또는 이의 등가 형태 행렬(표 2) 중 12개의 행(표 2의 블록 코드 관점에서는 열)을 천공하거나, 20개의 행(표 2의 블록 코드 관점에서는 열)을 선택하여 (20, 10) 블록 코드를 구성하는 것을 제안한다. 상기 표 1을 이용하는 경우와 상기 표 2를 이용하는 경우 모드 성능에 차이가 없는 바, 이하에서는 설명의 편의를 위해 특별한 표기가 없는 한 상기 표 2와 같은 TFCI 정보 코드의 등가 형태를 이용하는 것을 가정한다.

<71> 한편, 상기 TFCI 정보 부호화에 사용된 (32,10) 부호는 Reed-Muller(RM) 부호를 기반으로 생성되었다. 이 때, 오류 정정 성능을 위해 부호어의 최소거리($d_{\min}$)가 최대가 되는 천공 패턴을 찾는 것이 중요하다.

<72> 우선 본 실시형태와의 대비를 위해, TFCI 부호화에 사용된 (32,10) 부호의 생성행렬에 대해 최적의 천공패턴을 찾는 전수탐색(全數 探索; 전수탐색)을 수행하는 경우에 대해 알아보자. 32 * 10 행렬에서 천공되어야 할 생성행렬의 열(Column)의 개수를 p이라고 할 때, 모든 가능한 천공 패턴의 가지 수는 $\binom{32}{p}$이다. 여기서 $\binom{32}{p}$는 32개 중 p개를 선택하는 경우의 수를 나타낸다.

<73> 일례로 p = 12인 경우, $\binom{32}{12}$=225,792,840 개의 서로 다른 10 * 20 생성행렬이 존재하며, 상기 각각의 생성행렬당 이용하여 10 비트로 구성 가능한 2^{10} = 1,024 개의 정보가 20 비트의 부호어로 부호화한다. 상기 각각의 생성행렬로부터 생성되는 부호어들 간의 최소 해밍거리($d_{\min}$)을 계산하여, 상기 최소 해밍거리 중에서 가장 큰 값을 가지는 생성행렬을 찾는다. 상기 최대 $d_{\min}$ 을 갖는 생성행렬을 만들기 위해 사용된 천공 패턴이 최종적으로 찾고자 하는 패턴이다. 다만, 상술한 바와 같은 과정을 통한 최적의 (20, 10) 블록 코드의 생성은 너무나 많은 연산이 필요하여 바람직하지 않다.

<74> 따라서, 본 실시형태에서는 상술한 바와 같은 천공 패턴을 정하는데 있어서 특정 제약 조건을 가하여, 최적의 $d_{\min}$ 을 획득하기 위한 탐색 공간 (searching space)의 범위를 줄이는 방법을 제안하고자 한다.

<75> 이제 본 실시형태에 따라 $d_{\min} = d$ 를 갖는 부호어를 생성하는 (20,10) 부호의 생성행렬을 보다 효율적으로 찾는 방식에 대해 생각해 보자. 목표(Target) $d_{\min}$ 을 d라고 하면, (20,10) 부호의 생성행렬의 각 행 벡터(row vector) $g_{10by20}[i]$($1 \le i \le 10$)의 해밍 무게(Hamming weight) $w(g_{10by20}[i])$는 다음과 같은 필요조건을 갖는다.

<76> [수학식 1]

$$d \le w(g_{10by20}[i]) \le 20 - d$$
$$i = 0,1,\cdots,10$$

<77> $$i \ne 6(\textit{There are all ones in this row vector})$$

<78> 일례로, d=6 인 경우, 상기 수학식 1은 다음과 같이 나타낼 수 있다.

<79> [수학식 2]

<80> $$6 \le w(g_{10by20}[i]) \le 14$$

<81> 따라서, 본 실시형태에서는 상술한 전수 탐색 과정에서 생성되는 10 * 20 행렬의 각 행 벡터 $g_{10by20}[i]$들이 상기 수학식 2 의 제약을 갖도록 추가하면 d_{min} = 6 인 부호어를 생성하는 생성행렬을 찾기 위한 탐색 공간의 수(N $<<\binom{32}{12}$)를 감소시킬 수 있다. 일반적으로 문헌에 의하면, (20,10) 부호의 최대 d_{min} 은 6임이 알려져 있다. 이에 대한 구체적인 사항은 "The Theory of Error-Correcting Codes (by F.J. MacWilliams and N.J.A. Sloane)를 참조할 수 있다. 따라서, 본 실시형태에서는 상기 수학식 1의 조건에 d_{min} = 6 의 조건을 적용하여 최적의 성능을 나타내는 360개의 패턴을 찾았다.

<82> 이와 같은 360개의 천공 패턴은 본 발명자에 의해 발명되어 출원되었으며, 본 출원의 우선권 주장의 기초가 되는 미국 가출원 제 61/016,492 호 (GENERATION METHOD OF VARIOUS SHORT LENGTH BLOCK CODES WITH NESTED STRUCTURE BY PUNCTURING A BASE CODE)의 Appendix A에 상기 표 2를 기반으로 하여 천공되는 열의 인덱스가 일일이 개시되어 있으며, 본원 명세서에서는 지면 관계상 이를 생략한다.

<83> 본 발명에 대한 설명에 있어서는 상기 360개의 천공 패턴 중 다음과 같은 패턴을 예로 들어 설명한다.

<84> 아래 표 3은 상기 360개의 패턴 중 특정 해밍무게분포를 나타내는 패턴을 나타낸다.

<85> [표 3]

<86>

10X32 matrix	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
표2	0	0	1	1	1	0	1	1	0	0	1	1	1	1	0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1	0

<87> 상기 표 3의 천공 패턴은 상술한 본 출원의 우선권 주장의 기초가 되는 미국 가출원 제 61/016,492 호의 Appendix A, 표 A.2의 6번째 인덱스에 대응하는 천공 패턴이다. 이때, 상기 표 3에 있어서 "0" 은 해당 열을 천공하는 것을 나타내며, "1" 은 해당 열을 천공하지 않고 (20, 10) 블록 코드에 선택하는 것을 의미한다.

<88> 상기 표 3의 천공 패턴을 상기 표 2에 적용한 예는 다음과 같이 나타낼 수 있다.

<89> [표 4]

<90>

Index	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	puncturing for (20,10)
0	1	1	1	1	1	1	1	1	1	1	Punctured
1	0	1	1	1	1	1	1	1	1	0	Punctured
2	1	0	1	1	1	1	1	1	0	0	
3	0	0	1	1	1	1	0	0	1	0	
4	1	1	0	1	1	1	1	0	0	1	
5	0	1	0	1	1	1	1	0	1	0	Punctured
6	1	0	0	1	1	1	1	1	0	1	
7	0	0	0	1	1	1	0	1	0	0	
8	1	1	1	0	1	1	0	1	1	1	Punctured
9	0	1	1	0	1	1	0	0	1	1	Punctured
10	1	0	1	0	1	1	0	1	0	1	
11	0	0	1	0	1	1	0	1	1	1	
12	1	1	0	0	1	1	1	0	1	0	

13	0	1	0	0	1	1	1	1	0	1	
14	1	0	0	0	1	1	1	1	0	0	Punctured
15	0	0	0	0	1	1	1	0	0	0	
16	1	1	1	1	0	1	1	1	1	1	Punctured
17	0	1	1	1	0	1	1	0	0	1	
18	1	0	1	1	0	1	0	1	0	1	
19	0	0	1	1	0	1	0	1	1	0	
20	1	1	0	1	0	1	0	0	1	1	Punctured
21	0	1	0	1	0	1	1	0	1	1	
22	1	0	0	1	0	1	1	1	1	0	
23	0	0	0	1	0	1	0	1	1	0	Punctured
24	1	1	1	0	0	1	0	1	0	0	
25	0	1	1	0	0	1	0	0	1	0	
26	1	0	1	0	0	1	0	0	0	1	Punctured
27	0	0	1	0	0	1	1	0	1	1	
28	1	1	0	0	0	1	0	0	0	1	
29	0	1	0	0	0	1	1	0	0	0	Punctured
30	1	0	0	0	0	1	0	0	0	0	
31	0	0	0	0	0	1	0	0	0	0	Punctured

<91>

<92> 이때, 상기 표 4은 상기 표 2와 대비하여 행과 열의 방향이 변경되어 표현되었으나, 동일한 의미를 가지며, 우측에는 각각의 행 방향 시퀀스 중 천공되는 12개의 행을 표시하고 있다. 이에 따라 생성되는 (20, 10) 블록 코드는 다음과 같다.

<93> [표 5]

Index	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$
2	1	0	1	1	1	1	1	1	0	0
3	0	0	1	1	1	1	0	0	1	0
4	1	1	0	1	1	1	1	0	0	1
6	1	0	0	1	1	1	1	1	0	1
7	0	0	0	1	1	1	0	1	0	0
10	1	0	1	0	1	1	0	1	0	1
11	0	0	1	0	1	1	0	1	1	1
12	1	1	0	0	1	1	1	0	1	0
13	0	1	0	0	1	1	1	1	0	1

<94>

15	0	0	0	0	1	1	1	0	0	0
17	0	1	1	1	0	1	1	0	0	1
18	1	0	1	1	0	1	0	1	0	1
19	0	0	1	1	0	1	0	1	1	0
21	0	1	0	1	0	1	1	0	1	1
22	1	0	0	1	0	1	1	1	1	0
24	1	1	1	0	0	1	0	1	0	0
25	0	1	1	0	0	1	0	0	1	0
27	0	0	1	0	0	1	1	0	1	1
28	1	1	0	0	0	1	0	0	0	1
30	1	0	0	0	0	1	0	0	0	0

<95>

<96> 한편, 상기 표 4 또는 표 5의 행의 순서는 3GPP에서 사용되는 TFCI 부호화를 위한 행렬 순서와는 약간의 차이가 있다. 상술한 바와 같이 코딩 이론에 있어서 각 행들의 위치를 변경하는 경우에도 생성된 부호어들의 성능에는

차이가 없는바, 상기 표 4 또는 표 5의 행의 순서를 TFCI 부호 행렬과 같이 맞추면 다음 표 6과 같이 나타낼 수 있다.

<97> [표 6]

표4의 Index	TFCI Index	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	Puncturing for(20,10)
30	0	1	0	0	0	0	1	0	0	0	0	
29	1	0	1	0	0	0	1	1	0	0	0	Punctured
28	2	1	1	0	0	0	1	0	0	0	1	
27	3	0	0	1	0	0	1	1	0	1	1	
26	4	1	0	1	0	0	1	0	0	0	1	Punctured
25	5	0	1	1	0	0	1	0	0	1	0	
24	6	1	1	1	0	0	1	0	1	0	0	
23	7	0	0	0	1	0	1	0	1	1	0	Punctured
22	8	1	0	0	1	0	1	1	1	1	0	
21	9	0	1	0	1	0	1	1	0	1	1	
20	10	1	1	0	1	0	1	0	0	1	1	Punctured
19	11	0	0	1	1	0	1	0	1	1	0	

<98>

18	12	1	0	1	1	0	1	0	1	0	1	
17	13	0	1	1	1	0	1	1	0	0	1	
16	14	1	1	1	1	0	1	1	1	1	1	Punctured
14	15	1	0	0	0	1	1	1	1	0	0	Punctured
13	16	0	1	0	0	1	1	1	1	0	1	
12	17	1	1	0	0	1	1	1	0	1	0	
11	18	0	0	1	0	1	1	0	1	1	1	
10	19	1	0	1	0	1	1	0	1	0	1	
9	20	0	1	1	0	1	1	0	0	1	1	Punctured
8	21	1	1	1	0	1	1	0	1	1	1	Punctured
7	22	0	0	0	1	1	1	0	1	0	0	
6	23	1	0	0	1	1	1	1	1	0	1	
5	24	0	1	0	1	1	1	1	0	1	0	Punctured
4	25	1	1	0	1	1	1	1	0	0	1	
3	26	0	0	1	1	1	1	0	0	1	0	
2	27	1	0	1	1	1	1	1	1	0	0	
1	28	0	1	1	1	1	1	1	1	1	0	Punctured
0	29	1	1	1	1	1	1	1	1	1	1	Punctured
31	30	0	0	0	0	0	1	0	0	0	0	Punctured
15	31	0	0	0	0	1	1	1	0	0	0	

<99>

<100> 상술한 바와 같이 상기 표 6은 상기 표 4와 행의 순서만 변화가 있으며, 다른 사항은 완전히 동일하다. 이런 경우는 부화 이론에 의해서 부호특성은 완전히 동일한 특성을 갖게 된다. 상기 표 6과 같은 표현 방법의 이점은 (20,10)부호에서 (18,10)부호로 천공 시에 마지막 2비트를 천공하여 생성 가능한 이점이 있다.

<101> 다음으로, 상술한 (20, 10) 코드를 최대 (20, 14) 코드까지 확장하는 경우에 대해 살펴본다.

<102> 상술한 바와 같이 본 발명의 일 실시형태에 따라 생성되는 (20, A) 블록 코드는 3GPP LTE 시스템에서 채널 품질

정보를 나타내는 CQI가 PUCCH를 통해 전송되기 위한 채널 코딩 방법에 이용되는 것을 가정한다. 또한, 상술한 바와 같은 (20, 10) 코드 생성 단계에서는 상기 3GPP LTE 시스템에서 CQI 정보의 비트 수는 4비트부터 10비트까지 다양할 수 있다는 점을 감안하여, 최대 (20, 10) 블록 부호까지 제안한 것이다. 다만, MIMO의 경우에는 CQI의 정보 비트수가 10비트보다 더 커질 수도 있다. 하지만, 실제 CQI 전송량은 CQI의 생성 방식에 따라 결정되므로, 부호화 과정을 위해서는 대략 적으로 4부터 14비트 정도까지의 여러 정보 비트 수를 모두다 지원하는 방식을 고려해 보도록 한다,

<103> 따라서, 상술한 단계에서 고려한 (20, 10) 블록 코드에 정보 비트 수에 따라 열을 추가하여 최대 14비트까지 지원할 수 있는 (20, 14) 블록 코딩 방법에 대해 살펴본다.

<104> 이와 같이 추가되는 열을 전수탐색을 통해 찾기 위해서는 대단히 많은 연산을 수행하여야 한다. 따라서, 모든 경우에 대해 전수 탐색을 수행하는 것은 효율적이지 못하다.

<105> 본 단계에서는 상기 표 6의 6번째 열이 모두 1로 구성된 기본 시퀀스(Basis Sequence)임에 주목한다. 따라서, 추가되는 열이 최소거리 d를 만족해야 한다면, 최소한의 "0" 의 개수는 d보다 크거나 같아야 한다. 본 예에서는 "0" 의 개수가 부호어들 간의 최소거리가 되기 때문이다. 자세히 설명하면, 추가되는 열과 모두 1로 구성된 기존 6번째 열과의 차이가 곧 두 부호어간의 거리이고, 따라서 추가되는 열의 "0" 의 개수가 곧 부호어간의 거리와 일치하게 된다.

<106> 일반적으로 (20, 10) 코드에서 가능한 최대 최소거리가 6인 반면, 본 발명에서와 같은 방법으로 (20,10)에서 확장하여 구성된 (20, 11) 코드에서 가능한 최대한의 최소 거리는 4이다. 구체적으로 20 비트 부호어에서 다양한 정보 비트 수에 따른 최대 최소거리 특성은 다음과 같다.

<107> [표 7]

<108>

n \ k	4	5	6	7	8	9	10	11	12	13	14
20	8	8	8	6	6	6	6	4	4	4	4

<109> 따라서, 본 발명의 일 실시형태에서는 추가되는 열의 최대 최소거리가 "4" 인 것을 목표로 하여 열을 추가하는 방법을 제안한다. 이는 곧 추가되는 열에 "0" 의 개수가 4 이상인 열을 추가하는 것을 의미한다.

<110> 우선 탐색 개수를 최소화하기 위해 본 실시형태에서는 추가되는 열은 "0" 을 4개 포함하는 열로 한정하여 본다. 이와 같이 "0" 을 4개 포함하고, "1" 을 16개 포함하는 추가열은 여러 가지가 가능하며 다음 표 8은 그 중 일례로서 상기 표 6의 (20, 10) 코드를 (20, 14) 코드까지 확장한 형태를 나타내고 있다.

<111> [표 8]

<112>

I	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$	$M_{i,13}$
0	1	0	0	0	0	1	0	0	0	0	1	1	0	0
2	1	1	0	0	0	1	0	0	0	1	1	1	0	0
3	0	0	1	0	0	1	1	0	1	1	1	1	1	1
5	0	1	1	0	0	1	0	0	1	0	1	1	1	1
6	1	1	1	0	0	1	0	1	0	0	1	1	1	1
8	1	0	0	1	0	1	1	1	1	0	1	1	1	0
9	0	1	0	1	0	1	1	0	1	1	1	1	1	0
11	0	0	1	1	0	1	0	1	1	0	1	1	1	1
12	1	0	1	1	0	1	0	1	0	1	1	1	1	1
13	0	1	1	1	0	1	1	0	0	1	1	1	1	1

16	0	1	0	0	1	1	1	1	0	1	1	1	1	1
17	1	1	0	0	1	1	1	0	1	0	1	1	1	1
18	0	0	1	0	1	1	0	1	1	1	1	1	1	1
19	1	0	1	0	1	1	0	1	0	1	1	1	1	1
22	0	0	0	1	1	1	0	1	0	0	1	0	1	1
23	1	0	0	1	1	1	1	1	0	1	1	0	1	1
25	1	1	0	1	1	1	1	0	0	1	0	1	1	1
26	0	0	1	1	1	1	0	0	1	0	0	1	1	1
27	1	0	1	1	1	1	1	1	0	0	0	0	0	1
31	0	0	0	0	1	1	1	0	0	0	0	0	0	1

<113>

<114> 상기 표 8에 있어서 추가되는 4개 열은 우측 4개 열이며, 추가되는 열에 있어서 "0" 은 굵은 글씨체로 나타내었다.

<115> 이를 바탕으로 이하에서는 상기 표 8을 변형하거나, 최적화하는 방법에 대해 살펴본다.

<116> 본 발명의 일 실시형태에서는 상기 표 8의 6번째 열, 즉 $M_{i,5}$의 모든 비트가 "1" 인 기본 시퀀스(basis sequence)임을 주목한다. 이러한 기본 시퀀스는 해당 비트의 전체 부호어에 대한 기여도를 높이게 된다. 따라서 해당비트 관점으로 보면 이렇게 가중치(weight)가 많은 기본 시퀀스가 바람직할 수 있다.

<117> 하지만, 전체 부호어는 여러 비트가 "exclusive or" 로 연산되므로, 복합된 결과를 고려해야 한다. 따라서, 서로 결합되는 경우의 수를 줄이기 위해서, 상술한 바와 같이 모든 비트가 "1" 인 기본 시퀀스를 제일 앞으로 옮겨서 기여도를 높이는 방법이 고려가 가능하다. 이와 같은 방법으로, 비트수가 적은 경우의 부호화의 경우에 모든 비트가 1인 기본 시퀀스를 제일 앞으로 옮기는 방법이 고려가능하며, 아래 표 9에 나타내었다.

<118> [표 9]

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$	$M_{i,13}$
0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	0	0	0	0	1	1	1	0	0
2	1	0	0	1	0	0	1	0	1	1	1	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1	1	1	1
4	1	1	1	1	0	0	0	1	0	0	1	1	1	1
5	1	1	0	0	1	0	1	1	1	0	1	1	1	0
6	1	0	1	0	1	0	1	0	1	1	1	1	1	0
7	1	0	0	1	1	0	0	1	1	0	1	1	1	1
8	1	1	0	1	1	0	0	1	0	1	1	1	1	1
9	1	0	1	1	1	0	1	0	0	1	1	1	1	1
10	1	0	1	0	0	1	1	1	0	1	1	1	1	1
11	1	1	1	0	0	1	1	0	1	0	1	1	1	1
12	1	0	0	1	0	1	0	1	1	1	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1	1	1	1
14	1	0	0	0	1	1	0	1	0	0	1	0	1	1
15	1	1	0	0	1	1	1	1	0	1	1	0	1	1
16	1	1	1	0	1	1	1	0	0	1	0	1	1	1
17	1	0	0	1	1	1	0	0	1	0	0	1	1	1
18	1	1	0	1	1	1	1	1	0	0	0	0	0	1
19	1	0	0	0	0	1	1	0	0	0	0	0	0	1

<119>

<120> 상기 표 9는 원래 열 방향 기본 시퀀스 중 6번째 시퀀스, 즉 모두 "1" 을 포함하는 시퀀스 인덱스를 가장 첫 번

째 기본 시퀀스로 위치를 변경하고, 다른 기본 시퀀스들의 순서는 변경하지 않은 경우에 해당한다.

<121> 이하의 설명에 있어서 PUCCH 전송용 (20, 14) 블록 코드 구조로는 상기 표 9의 블록 코드를 이용하는 것을 가정한다. 만일 PUCCH 전송을 위해 정보 비트의 수가 최대 13비트 이하라고 한정되는 경우는 상기 표 9의 구조에서 가장 우측에 위치한 기본 시퀀스를 생략하고 나타낼 수 있으며, 이는 아래 표 10에 나타내었다.

<122> [표 10]

i	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$
0	1	1	0	0	0	0	0	0	0	0	1	1	0
1	1	1	1	0	0	0	0	0	0	1	1	1	0
2	1	0	0	1	0	0	1	0	1	1	1	1	1
3	1	0	1	1	0	0	0	0	1	0	1	1	1
4	1	1	1	1	0	0	0	1	0	0	1	1	1
5	1	1	0	0	1	0	1	1	1	0	1	1	1
6	1	0	1	0	1	0	1	0	1	1	1	1	1
7	1	0	0	1	1	0	0	1	1	0	1	1	1
8	1	1	0	1	1	0	0	1	0	1	1	1	1
9	1	0	1	1	1	0	1	0	0	1	1	1	1
10	1	0	1	0	0	1	1	1	0	1	1	1	1
11	1	1	1	0	0	1	1	0	1	0	1	1	1
12	1	0	0	1	0	1	0	1	1	1	1	1	1
13	1	1	0	1	0	1	0	1	0	1	1	1	1
14	1	0	0	0	1	1	0	1	0	0	1	0	1
15	1	1	0	0	1	1	1	1	0	1	1	0	1
16	1	1	1	0	1	1	1	0	0	1	0	1	1
17	1	0	0	1	1	1	0	0	1	0	0	1	1
18	1	1	0	1	1	1	1	1	0	0	0	0	0
19	1	0	0	0	0	1	1	0	0	0	0	0	0

<123>

<124> 이하에서는 이를 바탕으로 (n, k) 블록 코드 (n≤32, k≤14)를 생성하는 방법에 대해 살펴본다.

<125> **(n, k) 블록 코드 (n≤32, k≤14)**

<126> 이하의 설명에서는 최대 크기가 (32,14)인 블록 코드를 고려해보도록 한다. 즉, 본 실시형태에서는 부호화 비트수의 최대크기는 32이고, 정보비트수의 최대크기는 14인 경우를 다룬다. 다양한 방법으로 부호 설계가 가능하지만, 본 실시형태에서는 상술한 바와 같이 기존 부호와의 공통점을 최대한으로 찾는 방향으로 설계해보도록 한다.

<127> (32, 14) 블록 코드 생성을 위해 일단 기존 3GPP의 Release 99에서 사용하던 (32,10) TFCI 블록 코드로부터 얻어진 상기 표 9와 같은 (20,14) 블록 코드에 상술한 (32,10) TFCI 블록 코드까지 동시에 고려해 본다. 이들을 이용하여 (32, 14) 블록 코드를 생성하기 위해서는 다음 도 1에서 "TBD(To be defined)" 로 표시된 부분과 같은 부분에 대해 추가적인 정의가 필요하다.

<128> 도 1은 본 발명의 일 실시형태에 따라 기존 TFCI 정보 코딩에 이용된 (32, 10) 블록 코드와 PUCCH 전송용으로 이용되는 (20, 14) 코드를 이용하여 (32, 14) 블록 코드를 생성하는 개념도를 도시한 도면이다.

<129> 즉, 본 실시형태에서 (32, 14) 블록 코드(104) 생성을 위해 기존 TFCI 정보 코딩에 이용된 (32, 10) 블록 코드(101) 및 본 발명의 상술한 실시형태에서 제안된 (20, 14) 블록 코드(102)를 이용할 경우, TBD로 표기된 부분(103)만큼에 대한 추가적인 정의가 필요하다. 또한, 어느 한 블록 코드의 관점에서 보았을 때 이는 다양하게 해석될 수 있다. 즉, 본 실시형태에 따른 (32, 14) 블록 코드(104)는 기존 (32, 10) 블록 코드 우측에 4개의 기본 시퀀스(도 1에서 102a와 103의 결합 부분에 대응)를 추가하여 생성된 것으로 볼 수 있으며, 다른 관점에서는 상기 표 9 또는 이의 등가 형태의 (20, 14) 블록 코드(102)에 12개의 행 방향 시퀀스(도 1에서 101a와 103의 결합

부분에 대응)를 추가하여 생성된 것으로도 볼 수 있다.

<130> 이때, 상기 (32, 10) 블록 코드(101)는 상기 표 1에 나타낸 바와 같은 기존 TFCI 정보 코드 자체 또는 이들 내의 행간 그리고/또는 열간 위치를 변경한 등가 형태가 모두 이용가능하며, (20, 14) 블록 코드(102) 역시 상기 표 9에 나타낸 블록 코드 또는 이의 등가 형태가 이용될 수 있다.

<131> 한편, 상기 TBD(103)는 최소거리 관점에서 가장 좋은 성능을 나타내도록 부호를 설계하는 것이 바람직하다. 일반적으로 다양한 정보 비트 길이 및 부호화 비트 길이에 따른 최소거리 성능은 다음과 같다.

<132> [표 11]

<133>

n \ k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
20	20	10	8	8	8	8	6	6	6	6	4	4	4	4
32	32	16	16	16	16	16	12	12	12	12	10	10	8	8

<134> 상기 표 11로부터 (32, A) 블록 코딩에 있어서 A가 10보다 큰 경우, 기본 시퀀스의 최소 해밍거리의 최대 값은 10으로 제한됨을 알 수 있다.

<135> 따라서, 본 발명의 바람직한 일 실시형태에서는 도 1에서 TBD 부분(103)이, (20, 14) 블록 코드(102) 중 10비트 이상의 정보 비트에 대응하는 부분(102a)과 함께, 추가되는 각 기본 시퀀스(도 1의 102a와 103의 결합 부분에 대응)의 최소 해밍거리의 최대값이 10이 되는 조건을 만족하도록 규정되는 것을 제안한다. 구체적으로 추가되는 기본 시퀀스가 1개 또는 2개인 경우(즉, A가 11 또는 12인 경우)에는 각 기본 시퀀스의 최소 해밍거리가 10을 만족하도록 추가하는 것을 의미하며, 추가되는 기본 시퀀스가 3개 또는 4개인 경우(즉, A가 13 또는 14인 경우)에는 각 기본 시퀀스의 최소 해밍 거리가 8을 만족하도록 추가하는 것을 의미한다. 이는, 기존 TFCI 정보 코딩을 위한 (32, 10) 블록 코드에는 모든 성분이 "1" 인 기본 시퀀스를 포함하는바 이는 추가되는 각 기본 시퀀스가 "0" 을 10개 포함하는 경우로 해석될 수 있다. 아울러, 만일 (20, 14) 블록 코드를 상기 표 9와 같은 블록 코드로 이용하는 경우 도 1의 102a 부분에 대응하는 각 기본 시퀀스들이 "0" 을 4개 포함하고 있는 바, TBD 부분에 대응하는 기본 시퀀스 부분이 "0" 을 6개 포함하도록 설정하는 것으로 해석할 수도 있다.

<136> 상술한 바와 같은 조건을 만족하는 일례를 아래 표 12에 나타내었다.

<137> [표 12]

i	TFCI	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$	$M_{i,13}$
0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
1	2	1	1	1	0	0	0	0	0	0	1	1	1	0	0
2	3	1	0	0	1	0	0	1	0	1	1	1	1	1	1
3	5	1	0	1	1	0	0	0	0	1	0	1	1	1	1
4	6	1	1	1	1	0	0	0	1	0	0	1	1	1	1
5	8	1	1	0	0	1	0	1	1	1	0	1	1	1	0
6	9	1	0	1	0	1	0	1	0	1	1	1	1	1	0
7	11	1	0	0	1	1	0	0	1	1	0	1	1	1	1
8	12	1	1	0	1	1	0	0	1	0	1	1	1	1	1
9	13	1	0	1	1	1	0	1	0	0	1	1	1	1	1
10	16	1	0	1	0	0	1	1	1	0	1	1	1	1	1
11	17	1	1	1	0	0	1	1	0	1	0	1	1	1	1
12	18	1	0	0	1	0	1	0	1	1	1	1	1	1	1
13	19	1	1	0	1	0	1	0	1	0	1	1	1	1	1
14	22	1	0	0	0	1	1	0	1	0	0	1	0	1	1
15	23	1	1	0	0	1	1	1	1	0	1	1	0	1	1
16	25	1	1	1	0	1	1	1	0	0	1	0	1	1	1
17	26	1	0	0	1	1	1	0	0	1	0	0	1	1	1
18	27	1	1	0	1	1	1	1	1	0	0	0	0	0	1
19	31	1	0	0	0	0	1	1	0	0	0	0	0	0	1
20	1	1	0	1	0	0	0	1	0	0	0	1	0	1	1
21	4	1	1	0	1	0	0	0	0	0	1	1	1	0	0

<138>

22	7	1	0	0	0	1	0	0	1	1	0	1	0	1	1
23	10	1	1	1	0	1	0	0	0	1	1	1	1	0	1
24	14	1	1	1	1	1	0	1	1	1	1	0	0	1	0
25	15	1	1	0	0	0	1	1	1	0	0	1	1	1	0
26	20	1	0	1	1	0	1	0	0	1	1	0	0	0	1
27	21	1	1	1	1	0	1	0	1	1	1	0	1	0	0
28	24	1	0	1	0	1	1	1	0	1	0	0	1	0	1
29	28	1	0	1	1	1	1	1	1	1	0	0	1	1	0
30	29	1	1	1	1	1	1	1	1	1	1	1	0	1	0
31	30	1	0	0	0	0	0	0	0	0	0	0	0	0	1

<139>

<140> 상기 표 12에서 예를 들어 아래부터 12개의 행을 삭제하면 기존 PUCCH 전송용 (20,14) 부호가 되고, 아래부터 14개의 행을 삭제하면 (18,14) 부호가 된다. 한편, 본 실시형태에서 정보 비트의 가변적용은 쉽게 가능하며, 해당 정보 비트 수만큼의 상기 표 12에서 기본 시퀀스를 취하여 부호화에 이용하면 된다. 즉, 상기 표 12 에서 최대 정보 비트 수가 14비트보다 작은 비트 수만을 필요로 한다면, 아예 상기 표 12와 같은 기본 시퀀스 표에서 필요 없는 수만큼의 기본 시퀀스를 오른쪽 열부터 삭제하는 것도 가능하다. 이는 (32, 10) 블록 코드에 필요한 정보 비트 수만큼의 기본 시퀀스를 추가하는 것으로 해석할 수도 있다.

<141> 예를 들어, 최대 정보 비트 수가 11비트로 제한되는 경우를 가정하면 다음과 같은 (32, 11) 블록 코드를 이용할 수 있다.

<142> [표 13]

i	TFCI	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$
0	0	1	1	0	0	0	0	0	0	0	0	1
1	2	1	1	1	0	0	0	0	0	0	1	1
2	3	1	0	0	1	0	0	1	0	1	1	1
3	5	1	0	1	1	0	0	0	0	1	0	1
4	6	1	1	1	1	0	0	0	1	0	0	1

<143>

5	8	1	1	0	0	1	0	1	1	1	0	1
6	9	1	0	1	0	1	0	1	0	1	1	1
7	11	1	0	0	1	1	0	0	1	1	0	1
8	12	1	1	0	1	1	0	0	1	0	1	1
9	13	1	0	1	1	1	0	1	0	0	1	1
10	16	1	0	1	0	0	1	1	1	0	1	1
11	17	1	1	1	0	0	1	1	0	1	0	1
12	18	1	0	0	1	0	1	0	1	1	1	1
13	19	1	1	0	1	0	1	0	1	0	1	1
14	22	1	0	0	0	1	1	0	1	0	0	1
15	23	1	1	0	0	1	1	1	1	0	1	1
16	25	1	1	1	0	1	1	1	0	0	1	0
17	26	1	0	0	1	1	1	0	0	1	0	0
18	27	1	1	0	1	1	1	1	1	0	0	0
19	31	1	0	0	0	0	1	1	0	0	0	0
20	1	1	0	1	0	0	0	1	0	0	0	1
21	4	1	1	0	1	0	0	0	0	0	1	1
22	7	1	0	0	0	1	0	0	1	1	0	1
23	10	1	1	1	0	1	0	0	0	1	1	1
24	14	1	1	1	1	1	0	1	1	1	1	0
25	15	1	1	0	0	0	1	1	1	0	0	1
26	20	1	0	1	1	0	1	0	0	1	1	0
27	21	1	1	1	1	0	1	0	1	1	1	0
28	24	1	0	1	0	1	1	1	0	1	0	0
29	28	1	0	1	1	1	1	1	1	1	0	0
30	29	1	1	1	1	1	1	1	1	1	1	1
31	30	1	0	0	0	0	0	0	0	0	0	0

<144>

<145> 한편, 이하에서는 (16, k) 블록 코드까지 고려하여 (32, k) 블록 코드를 재배치하는 방법에 대해 살펴본다.

<146> 앞의 상기 표 12 또는 표 13의 (32,k) 블록 코드는 (20,k) 또는 (18,k) 블록 코드 생성시 맨 아래 행부터 남는 행수만큼 삭제하면 최적의 부호가 생성된다. 하지만, (16,k) 블록 코드가 필요한 경우에는 좀더 많은 고려가 필요하다. 따라서, 본 발명의 일 실시형태에서는 (16,k) 블록 코드까지 고려하여 상기 제안된 (32, k)블록 코드의 행의 순서를 바꾸어, (16, k) 블록 코드가 필요한 경우 (32, k) 블록 코드의 맨 아래 행부터 삭제하여 이용하더라도 부호가 생성되도록 설정하는 것을 제안하며, 그 일례를 아래 표 14에 나타내었다.

<147> [표 14]

i	TFCI	$M_{i,0}$	$M_{i,1}$	$M_{i,2}$	$M_{i,3}$	$M_{i,4}$	$M_{i,5}$	$M_{i,6}$	$M_{i,7}$	$M_{i,8}$	$M_{i,9}$	$M_{i,10}$	$M_{i,11}$	$M_{i,12}$	$M_{i,13}$
0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
1	2	1	1	1	0	0	0	0	0	0	1	1	1	0	0
2	3	1	0	0	1	0	0	1	0	1	1	1	1	1	1
3	5	1	0	1	1	0	0	0	0	1	0	1	1	1	1
4	6	1	1	1	1	0	0	0	1	0	0	1	1	1	1
6	9	1	0	1	0	1	0	1	0	1	1	1	1	1	0
8	12	1	1	0	1	1	0	0	1	0	1	1	1	1	1
9	13	1	0	1	1	1	0	1	0	0	1	1	1	1	1
10	16	1	0	1	0	0	1	1	1	0	1	1	1	1	1
11	17	1	1	1	0	0	1	1	0	1	0	1	1	1	1
12	18	1	0	0	1	0	1	0	1	1	1	1	1	1	1
13	19	1	1	0	1	0	1	0	1	0	1	1	1	1	1
15	23	1	1	0	0	1	1	1	1	0	1	1	0	1	1
16	25	1	1	1	0	1	1	1	0	0	1	0	1	1	1
17	26	1	0	0	1	1	1	0	0	1	0	0	1	1	1
19	31	1	0	0	0	0	1	1	0	0	0	0	0	0	1
7	11	1	0	0	1	1	0	0	1	1	0	1	1	1	1
14	22	1	0	0	0	1	1	0	1	0	0	1	0	1	1
5	8	1	1	0	0	1	0	1	1	1	0	1	1	1	0
18	27	1	1	0	1	1	1	1	1	0	0	0	0	0	1
20	1	1	0	1	0	0	0	1	0	0	0	1	0	1	1
21	4	1	1	0	1	0	0	0	0	0	1	1	1	0	0
22	7	1	0	0	0	1	0	0	1	1	0	1	0	1	1
23	10	1	1	1	0	1	0	0	0	1	1	1	1	0	1
24	14	1	1	1	1	1	0	1	1	1	1	0	0	1	0
25	15	1	1	0	0	0	1	1	1	0	0	1	1	1	0
26	20	1	0	1	1	0	1	0	0	1	1	0	0	0	1
27	21	1	1	1	1	0	1	0	1	1	1	0	1	0	0

<148>

28	24	1	0	1	0	1	1	1	0	1	0	0	1	0	1
29	28	1	0	1	1	1	1	1	1	1	0	0	1	1	0
30	29	1	1	1	1	1	1	1	1	1	1	1	0	1	0
31	30	1	0	0	0	0	0	0	0	0	0	0	0	0	1

<149>

<150> 상기 표 14는 (32,k) 부호에서 (20,k) 부호 생성시 맨 아래 행부터 12개의 행을 삭제하면 최적의 부호가 생성되고, (18,k) 부호는 맨 아래 행부터 14개의 행을, (16,k) 부호는 맨 아래 행부터 16개의 행을 삭제하면 최적의 부호가 생성되도록 설정한 예이다.

<151> 한편, 이하에서는 상술한 내용을 바탕으로 특정한 경우 32비트 이상의 부호어 길이가 필요한 경우의 채널 코딩 방법에 대해 살펴본다.

<152> **32비트 이상의 부호어가 필요한 경우**

<153> 먼저, 본 발명의 일 실시형태에서는 32 비트 이상의 부호어가 필요한 경우, (32,k) 부호 또는 (20,k) 부호를 기본 부호로 하여 상기 기본 부호를 반복하여 긴 길이의 (n,k) 부호를 생성하는 방법을 제안한다.

<154> 앞의 표 12 또는 표 14를 통해 (32,k) 부호와 (20,k) 부호의 생성은 쉽게 가능하다. 한편, 전송비트에 좀더 강력한 오류정정 능력을 가지도록 하기 위해서 부호화 비트의 수를 늘리는 경우를 고려해 보자. 이때, 늘어난 부호화 비트 수에 해당하는 새로운 부호를 생성하는 것이 바람직하지만, 실제 매번 새로운 부호를 설계하는 것은 많은 제약이 따른다. 따라서 간단한 생성 방법 중의 하나는 기본부호를 원하는 길이만큼 반복하는 것이 고려 가

능하다. 만일 원하는 길이가 기본부호의 정수 배로 정확하게 일치하지 않는 다면, 원하는 길이보다 크게 반복한 후, 초과 비트 수만큼을 제거하는 방법이 고려 가능하다. 이때, 최적의 천공패턴을 매번 찾는 것도 가능하지만, 레이트 매칭 블록(rate matching block) 등을 사용하여 간단하게 천공하는 방법도 실용적으로 고려 가능하다.

<155> 여기서 기본부호는 (32,k)를 고려할 수도 있고, (20,k)를 고려할 수도 있다. 편의상 기본부호의 정수 배 크기를 갖는 경우만 고려하고, 그 이외의 경우에는 천공을 통해 해결하는 것으로 가정한다. 또한, k값은 여러 개가 가능하므로, 여기서는 최대 크기인 14를 기준으로 설명하며, 14보다 작은 크기는 특별히 언급하지는 않지만, 상술한 바와 같이 해당 길이만큼 기본 시퀀스를 선택하여 이용할 수 있음은 상술한 설명을 통해 당업자에게 자명하다.

<156> 예를 들어, (64,14) 부호가 필요한 경우, (32,14) 부호를 단순히 두 번 반복하는 것이 고려 가능하다. 또한, (40,14) 부호가 필요한 경우에는 (20,14) 부호를 단순히 두번 반복하는 것이 고려 가능하다. 또한, 기본 부호로 (32,14) 부호와 (20,14) 부호를 동시에 고려하여, (52,14) 부호의 경우에는 (32,14) 부호와 (20,14) 부호를 연달아 붙여서 구성하는 것이 가능하다.

<157> 결국 본 실시형태에 따라 가능한 부호의 조합은 (a*32+b*20,14) 부호가 생성 가능하다(여기서 a와 b는 0이상의 정수 값이다).

<158> 만일 최종적으로 필요한 부호화 비트수가 기본부호의 정수배가 아닌 경우에는 상술한 바와 같이 원하는 길이보다 크도록 반복한 후에 끝에서부터 잘라내거나, 레이트 매칭 블록 등을 통한 천공 이 고려 가능하다.

<159> 한편, 본 발명의 다른 일 실시형태에서는 정보 비트의 순서를 역순으로 한 후, 기본 부호를 반복하는 방법을 고려한다.

<160> 앞에서 살펴본 대로 기본부호를 계속적으로 반복하는 경우에는 기본부호의 최소거리 특성이 그대로 유지된 채로 반복된다. 따라서, 원래 최소거리가 4인 부호를 두 번 반복하면 단순히 최소거리가 2배로 늘어나서 8로 변하게 된다.

<161> 하지만, 반복 시에 정보비트에 변화를 주게 되면, 첫 번째의 경우에 최소거리를 갖는 부호어를 만들어내는 정보비트라도, 반복 시에 변화된 정보비트에 의해 생성된 부호어는 최소거리보다 큰 거리를 갖는 부호어를 만들어낼 수 있다. 이와 같은 원리로 최소거리가 반복되지 않도록 한다면 최소거리 특성이 단순 배수보다는 커지도록 부호를 구성하는 것이 가능하다.

<162> 상기 정보 비트의 변화의 방법으로는 여러 가지가 고려 가능하다. 즉, 비트단위에서 반전(reverse)을 취하는 방법, 정보 비트를 PN 시퀀스 등과 같은 랜덤 시퀀스(random sequence)에 통과시켜서 변화를 주는 방법 등이 고려 가능하다.

<163> 또한, 매번 반복 시마다 정보비트의 변화를 각각 다르게 주는 방법이 고려 가능하다. 하지만, 이렇게 되면 송수신단에서 복잡도가 증가하게 되므로, 여기서는 반복이 여러 번 되는 경우에, 한번은 그대로 다음 번은 정보비트에 변화를 주는 경우를 고려해 본다.

<164> 좀더 자세히 설명하면, 부호의 반복시 짝수번째는 정보비트를 그대로 두고, 홀수번째에는 정보비트에 변화를 주는 방식을 고려해보도록 한다. 즉, 정보비트의 변화가 반복회수마다 토글(toggle) 방식으로 변화가 가해지는 경우를 고려한다.

<165> 여기서는 간단한 예를 (20,k) 부호와 (32,k) 부호를 기본 부호로 하고, (40,k) 부호와 (52,k) 부호 그리고 (64,k) 부호를 생성하는 경우를 고려해보자.

<166> 도 2는 본 발명의 일 실시형태에 따라 (20, k) 블록 코드와 (32, k) 블록 코드를 기본 부호로 하여, (40, k), (52, k) 및 (64, k) 블록 코드를 생성하는 경우의 최소거리 성능을 나타낸 그래프이다.

<167> 도 2에서 "(40,k)_20+20" 는 기본 부호인 (20,k) 부호를 그냥 2번 반복한 경우를 의미한다. 그리고, "(20,k)_20rev" 은 기본부호인 (20,k) 부호에서 정보비트를 비트 반전하여 적용한 경우를 의미한다. "(32,k)_32rev" 도 같은 방법으로 해석 가능하다.

<168> 또한, "(40,k)_20+20rev" 는 기본부호로 (20,k)부호를 고려하여 2번 반복하는데 있어서, 토글 방식으로 정보비트에 변화를 주며, 2번째 반복 시에 정보비트를 비트 반전하여서 적용한 경우를 나타낸다. "(64,k)_32+32rev" 도 같은 방법으로 해석 가능하다. 한편, "(52,k)_20+32rev" 표시는 기본부호로 (32,k) 부호와 (20,k) 부호를 선택하여 상기 기본부호를 각각 한번씩 반복하지만, 두 번째 기본부호인 (32,k) 부호는 비트 반전하여 정보비트를

적용한 경우를 의미한다.

<169> 상기 도 2 에서 "(40,k)_20" 부호는 k≥4인 경우에 오히려 (32,k) 부호보다도 좋지 못한 성능을 나타낸다. 따라서, 단순히 (20,k)를 두 번 반복하기 보다는 차라리 (32,k) 부호를 사용하는 것이 적은 부호화 비트를 사용하고도 좋은 성능을 보임을 알 수 있다. 즉, (20,k) 부호를 반복하는 것 보다는 (32,k) 부호를 반복하는 것이 적은 부호화 비트를 사용하고도 좋은 성능을 보임을 알 수 있다, 따라서, 본 발명의 바람직한 일 실시형태에서는 실제 긴 길이의 부호화 비트를 위한 부호화 시에 (32,k) 부호를 반복하여 사용하는 것을 제안한다. 즉, 본 실시형태에 있어서 기본 부호로 (32,k) 부호를 선택하고, 최종적으로 필요한 출력 시퀀스의 비트수가 32비트 이상이며 기본부호의 정수배가 아닌 경우에는 원하는 길이보다 크도록 반복한 후에 끝에서부터 잘라내는 방법(즉, 필요한 출력 시퀀스의 비트수만큼 만을 선택하는 방법)을 사용하는 것이 가능하다.

<170> 상술한 바와 같은 실시형태는 다음과 같이 해석될 수도 있다. 즉, 필요한 출력 시퀀스의 길이가 32비트 이상인 경우 출력 시퀀스는 (32, k) 블록 코드에 의해 코딩된 결과를 순환적으로 반복하여 획득되는 것으로 볼 수 있다. 즉, (32, k) 블록 코드에 의해 채널 코딩된 32비트 길이의 부호어를 $b_0, b_1, b_2, b_3, \ldots, b_{B-1}$ (단, B=32)로 나타내고, 32보다 긴 길이(예를 들어, "Q" 비트 길이)의 출력 시퀀스를 $q_0, q_1, q_2, q_3, \ldots, q_{Q-1}$ 로 나타낼 경우, 출력 시퀀스와 채널 코딩된 32비트 길이 부호어간에는 다음과 같은 관계를 만족하는 것으로 볼 수 있다.

<171> [수학식 3]

<172> $$q_i = b_{(i \bmod B)} \quad \text{where} \quad i = 0, 1, 2, \ldots, Q-1$$

<173> 상기 수학식 3은 인덱스 i를 가지는 출력 시퀀스 성분이 인덱스 i를 32에 해당하는 B로 모듈로 연산을 수행한 결과값을 인덱스로 가지는 부호어 성분과 대응하도록 하며, Q가 32 보다 클 경우 출력 시퀀스는 채널 코딩된 결과 시퀀스를 순환적으로 반복하여 획득하는 구조를 가짐을 알 수 있다. 이는 상술한 실시형태에서 (32, k) 부호를 소정 횟수 반복한 후, 필요한 부호어 길이만큼 선택하여 이용하는 경우와 동일한 의미를 가지며, 다만 이를 다른 관점에서 해석한 것이 불과함은 당업자에게 자명하다.

<174> 한편, 상기 도 2에서 "(52,k)_20+32" 보다는 "(52,k)_20+32rev" 의 성능이 같거나 좋으며, "(40,k)_20+20" 보다는 "(40,k)_20+20rev" 의 성능이 좋거나 같다. 따라서, 반복 시에 한번은 그대로 다음 번은 정보비트를 반전하여 적용하는 것이 최소거리 특성을 좋게 하는 바람직한 방법일 수 있다.

<175> 만일 최종적으로 필요한 정보 비트 수가 기본부호의 정수배가 아닌 경우에는 원하는 길이보다 크도록 반복한 후에 끝에서부터 잘라내거나, 레이트 매칭 블록 등을 통한 천공이 고려 가능하다.

<176> 한편, 기본 부호의 반복 시 토글 방식으로 정보 비트에 비트 반전을 적용하는 방법은 상기 언급된 기본 부호뿐만 아니라 다양한 기본부호의 반복에도 적용 가능하다. 예를 들어 ACK/NACK과 같은 제어정보의 심플랙스 부호 반복 부호화시에도 한번은 그대로, 다음 번에는 ACK/NACK 제어 정보의 비트 반전된 정보 비트를 부호화하여 반복하는 것이 가능하다.

<177> 상술한 바와 같이 개시된 본 발명의 바람직한 실시형태에 대한 상세한 설명은 당업자가 본 발명을 구현하고 실시할 수 있도록 제공되었다. 상기에서는 본 발명의 바람직한 실시 형태를 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

<178> 따라서, 본 발명은 여기에 나타난 실시형태들에 제한되려는 것이 아니라, 여기서 개시된 원리들 및 신규한 특징들과 일치하는 최광의 범위를 부여하려는 것이다.

산업이용 가능성

<179> 상술한 바와 같은 본 발명의 각 실시형태에 따른 채널 코딩 방법은 3GPP LTE 시스템에서 CQI/PMI 정보를 PUSCH를 통해 상향링크로 전송 시 수행하는 채널 코딩에 적용되기에 적합하다. 그러나, 상술한 방법들은 이와 같은 3GPP LTE 시스템에 한정되어 적용될 필요는 없으며, 다양한 길이를 가질 수 있는 정보에 블록 코딩을 수행하는 임의의 통신 방식에 동일한 원리에 따라 적용될 수 있다.

도면의 간단한 설명

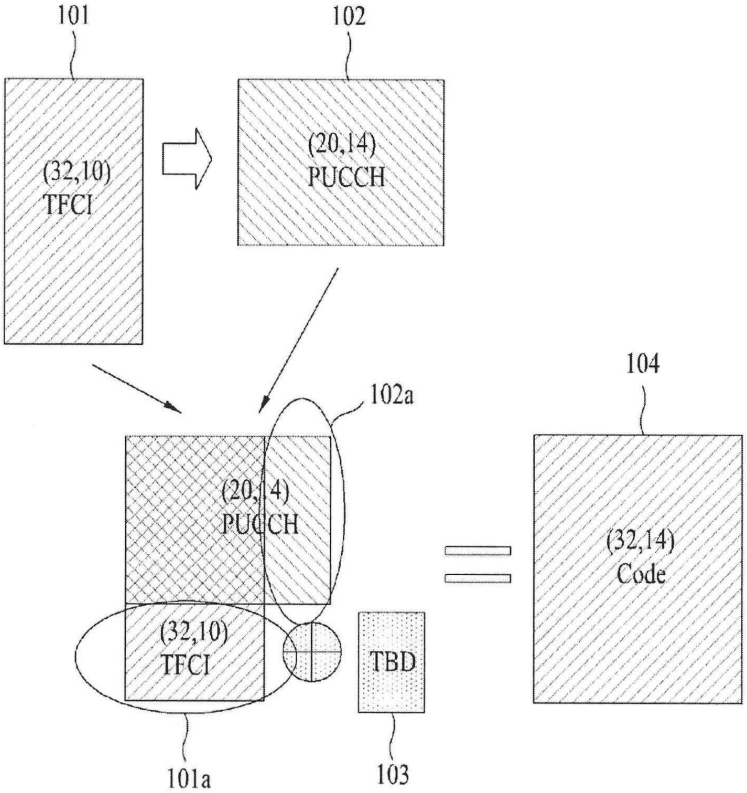
<180> 도 1은 본 발명의 일 실시형태에 따라 기존 TFCI 정보 코딩에 이용된 (32, 10) 블록 코드와 PUCCH 전송용으로

이용되는 (20, 14) 코드를 이용하여 (32, 14) 블록 코드를 생성하는 개념도를 도시한 도면이다.

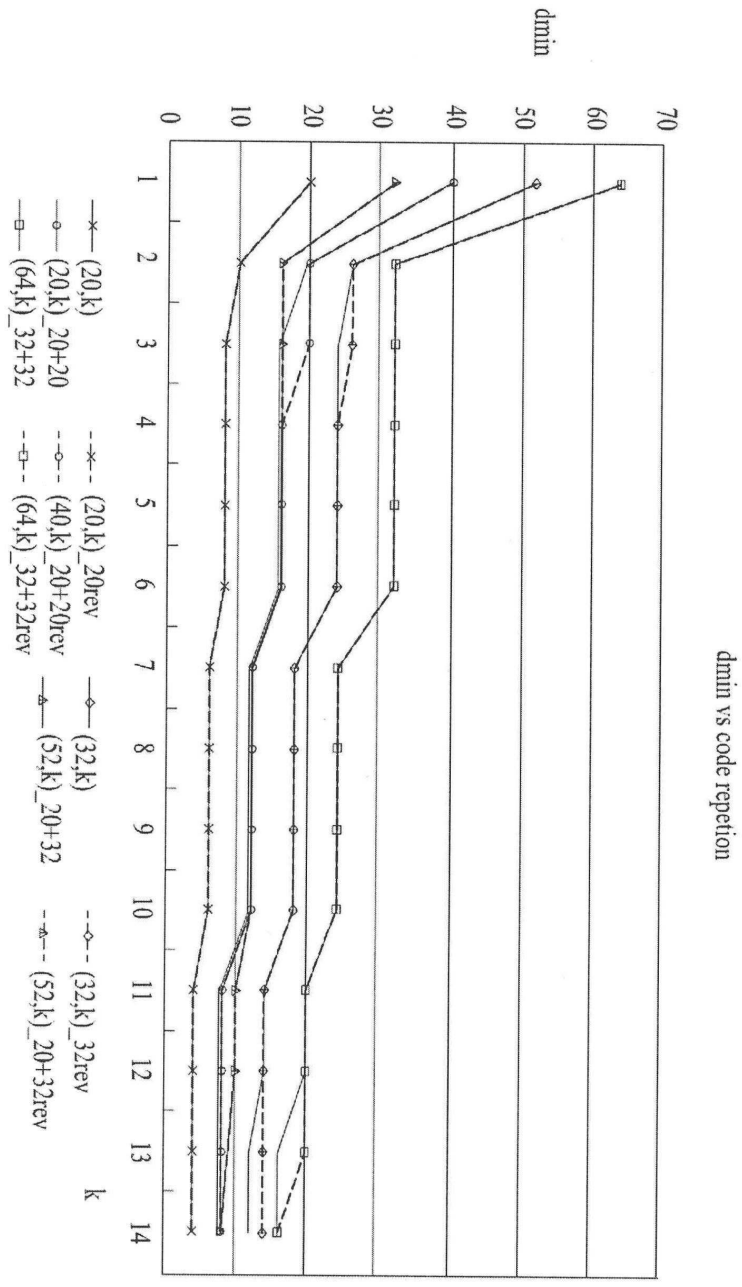
<181> 도 2는 본 발명의 일 실시형태에 따라 (20, k) 블록 코드와 (32, k) 블록 코드를 기본 부호로 하여, (40, k), (52, k) 및 (64, k) 블록 코드를 생성하는 경우의 최소거리 성능을 나타낸 그래프이다.

도면

도면1



도면2


dmin vs code repetion
dmin
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14
k
(20,k)
(20,k)_20+20
(64,k)_32+32
(20,k)_20rev
(40,k)_20+20rev
(64,k)_32+32rev
(32,k)
(52,k)_20+32
(32,k)_32rev
(52,k)_20+32rev