(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0016556
(43) 공개일자 2010년02월12일

(51) Int. Cl.
H04L 1/00 (2006.01)
(21) 출원번호 10-2009-7023798
(22) 출원일자 2008년05월16일
심사청구일자 없음
(85) 번역문제출일자 2009년11월13일
(86) 국제출원번호 PCT/US2008/006332
(87) 국제공개번호 WO 2008/144002
국제공개일자 2008년11월27일
(30) 우선권주장
60/930,591 2007년05월17일 미국(US)
(뒷면에 계속)

(71) 출원인
톰슨 라이센싱
프랑스 에프-92100 블로뉴-빌랑꾸르 꿰 아 르 갤로 46
(72) 발명자
시타, 리차드, 더블유.
미국 60302 일리노이주 오크 파크 노쓰 콜럼비안 739
로프레스토, 스콧 엠.
미국 60622 일리노이주 시카고 노쓰 오클리 블러바드 아파트먼트 넘버1 1329
(74) 대리인
양영준, 백만기, 전경석

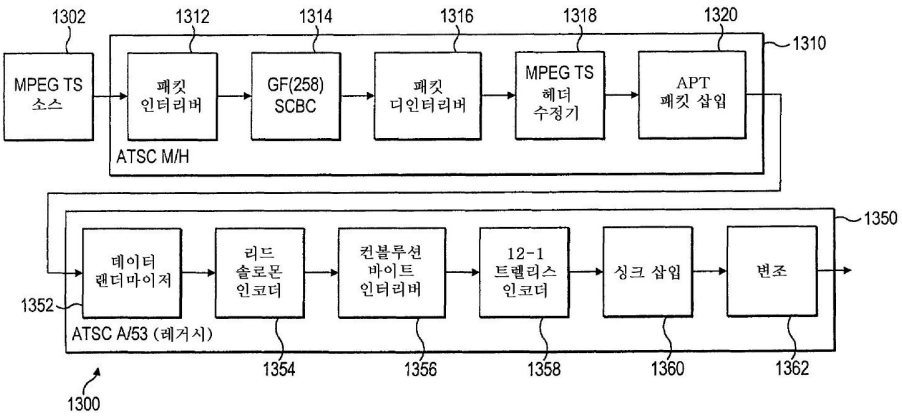
전체 청구항 수 : 총 31 항

(54) 신호를 인코딩 및 디코딩하는 장치 및 방법

(57) 요 약

종래의 방송 전송이 모바일 장치에 이용가능하게 하는 새로운 기술이 제공된다. 데이터 페이로드 및 헤더를 갖는 데이터의 패킷을 수신하는 단계, 패킷 내의 데이터를 바이트 코드 인코딩하는 단계(1740), 및 바이트 코드 인코딩하는 단계에 응답하여 상기 헤더 내의 정보를 변경하는 단계(1760)를 포함하는 방법(1700)이 설명된다. 데이터의 패킷을 수신하고 바이트 코드 인코딩 프로세스를 사용하여 데이터를 인코딩기 위한 인코더(1314) 및 바이트 코드 인코딩에 응답하여 헤더 내의 정보를 변경하기 위한 헤더 수정기(1318)를 포함하는 장치(1300)가 설명된다. 복수의 데이터 패킷을 수신하고 헤더 내의 정보에 기초하여 데이터 패킷을 식별하는 패킷 식별자, 식별된 데이터 패킷을 바이트 코드 디코딩 프로세스를 사용하여 디코딩하기 위한 바이트 코드 디코더(2006), 및 적어도 디코딩된 식별된 데이터 패킷을 리드-솔로몬 디코딩 프로세스를 사용하여 디코딩하기 위한 리드-솔로몬 디코더(2008)를 포함하는 장치(2000)가 설명된다.

대 표 도



(30) 우선권주장
60/930,683 2007년05월16일 미국(US)
60/936,360 2007년06월20일 미국(US)
60/958,581 2007년07월06일 미국(US)

특허청구의 범위

청구항 1

데이터 페이로드(payload) 및 헤더를 갖는 데이터의 패킷을 수신하는 단계(1710);

상기 패킷 내의 데이터를 바이트 코드 인코딩하는 단계(1740); 및

상기 바이트 코드 인코딩(1740)에 응답하여 상기 헤더 내의 정보를 변경하는 단계(1760)

를 포함하는 방법(1700).

청구항 2

제1항에 있어서,

상기 변경 단계(1760)는 상기 패킷이 레거시(legacy) 방송 인코딩된 패킷만을 수신할 수 있는 수신기에 의해서는 인식되지 않도록 상기 패킷을 변경하는 방법(1700).

청구항 3

제1항에 있어서,

상기 변경 단계(1760)는 상기 헤더 내의 프로그램 식별자를 변경하는 단계를 포함하는 방법(1700).

청구항 4

제3항에 있어서,

상기 변경 단계(1760)는 상기 프로그램 식별자를 레거시 프로그램 맵 테이블에서 사용되지 않는 값으로 설정하는 단계를 포함하는 방법(1700).

청구항 5

제4항에 있어서,

상기 변경 단계(1760)는 상기 프로그램 식별자를 유효 모바일 방송 패킷에 대한 값으로 설정하는 단계를 포함하는 방법(1700).

청구항 6

제1항에 있어서,

상기 바이트 코드 인코딩 단계(1740)는 갈루아 필드 생성 행렬(Galois Field generator matrix)을 사용하여 상기 패킷 내에 데이터를 보충(supplement)하는 방법(1700).

청구항 7

제1항에 있어서,

상기 수신 단계에서의 상기 패킷은 상기 데이터 페이로드에는 184 바이트의 데이터를 포함하고 상기 헤더에는 3 바이트의 데이터를 포함하는 방법(1700).

청구항 8

제1항에 있어서,

상기 바이트 코드 인코딩 단계(1740)는 GF(256) 직렬 연쇄 블록 코딩 프로세스를 사용하는 방법(1700).

청구항 9

제1항에 있어서,

상기 바이트 코드 인코딩 단계(1740)는 상기 패킷의 데이터 페이로드 부분만을 바이트 코드 인코딩하는 것을 포함하는 방법(1700).

청구항 10

제1항에 있어서,

상기 변경된 헤더를 포함하는 바이트 코드 인코딩된 데이터를 리드-솔로몬(Reed-Solomon) 인코딩하는 단계(1790)를 더 포함하는 방법(1700).

청구항 11

제1항에 있어서,

상기 방법은 텔레비전 방송 시스템에서 사용되는 방법(1700).

청구항 12

데이터 페이로드 및 헤더를 갖는 데이터의 패킷을 수신하기 위한 인코더(1314) - 상기 인코더는 바이트 코드 인코딩 프로세스를 사용하여 상기 데이터의 패킷 내의 데이터를 인코딩함 -; 및

상기 인코더(1314)에 결합되고, 상기 인코딩된 데이터의 패킷을 수신하고 상기 바이트 코드 인코딩에 응답하여 상기 헤더 내의 정보를 변경하기 위한 헤더 수정기(1318)

를 포함하는 장치(1300)

청구항 13

제12항에 있어서,

상기 헤더 수정기(1318)는 상기 패킷이 레거시 방송 인코딩된 패킷만을 수신할 수 있는 수신기에 의해서는 인식되지 않도록 상기 패킷을 변경하는 장치(1300).

청구항 14

제12항에 있어서,

상기 헤더 수정기(1318)는 프로그램 식별자를 레거시 프로그램 맵 테이블에서 사용되지 않는 값으로 설정함으로써 상기 헤더 내의 정보를 변경하는 장치(1300).

청구항 15

제12항에 있어서,

상기 인코더(1314)에서의 바이트 코드 인코딩 프로세스는 갈루아 필드 생성 행렬을 사용하여 상기 패킷 내에 데이터를 보충하는 장치(1300).

청구항 16

제12항에 있어서,

상기 수신된 데이터 패킷은 상기 데이터 페이로드에는 184 바이트의 데이터를 포함하고 상기 헤더에는 3 바이트의 데이터를 포함하는 장치(1300).

청구항 17

제12항에 있어서,

상기 인코더(1314)에서의 바이트 코드 인코딩은 GF(256) 직렬 연쇄 블록 코딩 프로세스를 사용하는 장치(1300).

청구항 18

제12항에 있어서,

상기 헤더 수정기(1318)에 결합된 리드-솔로몬 인코더(1354)를 더 포함하고, 상기 리드-솔로몬 인코더는 상기 변경된 헤더를 포함하는 상기 바이트 코드 인코딩된 데이터를 인코딩하는 장치(1300).

청구항 19

데이터 페이로드 및 헤더를 갖는 데이터의 패킷을 제공하기 위한 수단(1302);

상기 패킷 내의 데이터를 바이트 코드 인코딩하기 위한 수단(1314); 및

상기 바이트 코드 인코딩에 응답하여 상기 헤더 내의 정보를 변경하기 위한 수단(1318)

을 포함하는 장치(1300).

청구항 20

제19항에 있어서,

상기 변경 수단(1318)은 상기 패킷이 레거시 방송 인코딩된 패킷만을 수신할 수 있는 수신기에 의해서는 인식되지 않도록 상기 패킷을 변경하는 장치(1300).

청구항 21

제19항에 있어서,

상기 변경 수단(1318)은 프로그램 식별자를 레거시 프로그램 맵 테이블에서 사용되지 않는 값으로 설정하는 것을 포함하는 장치(1300).

청구항 22

제21항에 있어서,

상기 변경 수단(1318)은 상기 프로그램 식별자를 유효 모바일 방송 패킷에 대한 값으로 설정하는 것을 포함하는 장치(1300).

청구항 23

제19항에 있어서,

상기 바이트 코드 인코딩 수단(1314)은 갈루아 필드 생성 행렬을 사용하여 상기 패킷 내에 데이터를 보충하는 장치(1300).

청구항 24

제19항에 있어서,

상기 바이트 코드 인코더(1314)는 GF(256) 직렬 연쇄 블록 코딩 프로세스를 사용하는 장치(1300).

청구항 25

복수의 데이터 패킷을 수신하기 위한 패킷 식별자 - 상기 데이터 패킷 각각은 데이터 페이로드 및 헤더를 갖고, 상기 패킷 식별자는 상기 헤더 내의 정보에 기초하여 상기 복수의 패킷으로부터 데이터 패킷을 식별함 -,

상기 패킷 식별자에 결합되고, 상기 복수의 데이터 패킷을 수신하고 바이트 코드 디코딩 프로세스를 사용하여 상기 식별된 데이터 패킷을 디코딩하기 위한 바이트 코드 디코더(2006), 및

상기 바이트 코드 디코더에 결합되고, 리드-솔로몬 디코딩 프로세스를 사용하여 적어도 상기 디코딩된 식별된 데이터 패킷을 디코딩하기 위한 리드-솔로몬 디코더(2008)

를 포함하는 장치(2000).

청구항 26

제25항에 있어서,

상기 복수의 데이터 패킷 내의 상기 식별된 데이터의 패킷은 모바일 방송 데이터 패킷인 장치(2000).

청구항 27

제25항에 있어서,

상기 바이트 코드 디코딩 프로세스는 GF(256) 직렬 연쇄 블록 코드 디코딩 프로세스인 장치(2000).

청구항 28

각각 데이터 페이로드 및 헤더를 갖는 복수의 데이터 패킷을 수신하는 단계,

상기 헤더 내의 정보에 기초하여 상기 복수의 패킷으로부터 데이터 패킷을 식별하는 단계,

상기 식별된 데이터 패킷을 바이트 코드 디코딩 프로세스를 사용하여 디코딩하는 단계, 및

적어도 상기 디코딩된 식별된 데이터 패킷을 리드-솔로몬 디코딩 프로세스를 사용하여 디코딩하는 단계

를 포함하는 방법.

청구항 29

제28항에 있어서,

디코딩된 조합 데이터를 처리하여 모바일 수신기를 위한 데이터 스트림을 생성하는 단계를 더 포함하는 방법.

청구항 30

제28항에 있어서,

상기 식별된 데이터 패킷은 모바일 방송 데이터 패킷인 방법.

청구항 31

제28항에 있어서,

상기 바이트 코드 디코딩 프로세스는 갈루아 필드(256) 직렬 연쇄 블록 코드 디코딩 프로세스인 방법.

명 세 서

기 술 분 야

[0001] 본 출원은 2007년 5월 16일에 출원된 미국 가출원 60/930683, 2007년 5월 17일에 출원된 미국 가출원 60/930591, 2007년 6월 20일에 출원된 미국 가출원 60/936360, 및 2007년 7월 6일에 출원된 미국 가출원 60/958581의 35 U.S.C. 119조에 따른 우선권 주장 출원이다.

[0002] 본 발명은 디지털 방송 시스템의 동작에 관한 것으로, 특히 이동용, 보행자용 및 개인용 장치에서의 사용을 위한 방송 텔레비전용 데이터의 인코딩 및 디코딩에 관한 것이다.

배 경 기 술

[0003] 본 절은 다음에서 설명되는 본 발명의 다양한 양태에 관련될 수 있는 당해 기술분야의 다양한 양태에 대해 소개하기 위한 것이다. 본 논의는 본 발명의 다양한 양태의 이해를 용이하게 하는 배경 정보를 제공하는 데 도움이 될 것이다. 따라서, 본 설명은 이러한 관점에서 이해되어야 하는 것이지 종래 기술의 인정으로 이해되지 않아야 한다.

[0004] 전 세계의 텔레비전 방송 시스템은 아날로그 오디오 및 비디오 신호의 전달에서 현대 디지털 통신 시스템으로 바뀌었다. 예컨대, 미국에서, ATSC (Advanced Television Standards Committee)는 "ATSC 표준: 디지털 텔레비전 표준 A/53" (A53 표준)이라고 하는 표준을 개발하였다. A53 표준은 디지털 텔레비전 방송 데이터가 인코딩되고 디코딩되는 방식을 정의한다. 또한, 미국 FCC (Federal Communications Commission)는 텔레비전 방송을 위해 전자기 스펙트럼의 일부를 할당하였다. FCC는 할당 부분 내에서 연속해 인접하는(contiguous) 6MHz 채널을 육상 (즉, 케이블이나 위성이 아닌) 디지털 텔레비전 방송의 전달을 위한 방송사에게 할당한다. 각 6MHz 채널은 A53 표준에서 인코딩 및 변조 포맷에 기초하여 대략 19Mb/초의 채널 용량을 갖는다. 또한, FCC는 6MHz 채

널을 통한 육상 디지털 텔레비전 데이터의 전송은 A53 표준에 따라야 한다고 요구하였다.

[0005] A53 표준은 소스 데이터 (예컨대, 디지털 오디오 및 비디오 데이터)가 채널을 통해 전송되는 신호가 되도록 처리되어 변조되는 방식에 대해 정의한다. 이 처리는 리던던트(redundant) 정보를 소스 데이터에 추가하여 채널로부터 신호를 수신하는 수신기가 채널이 전송 신호에 노이즈와 다중경로 간섭을 부가하더라도 소스 데이터를 복구할 수 있도록 한다. 소스 데이터에 추가된 리던던트 정보는 소스 데이터가 전송되는 유효 데이터 속도를 감소시키지만 전송 신호로부터 소스 데이터의 성공적인 복구 가능성을 증가시킨다.

[0006] 도 1은 A53 표준에 따르는 신호를 전송하는 일반적인 전송 시스템(100)의 블록도이다. 데이터는 전송 소스(102)에 의해 생성되어 패킷으로 배열된다. 패킷은 크기가 187 바이트이고 하나 또는 그 이상의 코드워드를 포함할 수 있다. 각 패킷은 3바이트 헤더를 포함하는데 이 중 13비트는 패킷으로 보내지는 데이터의 종류를 식별하는 패킷 ID (PID)이다. 예컨대, 0x11 (hex 11) 값을 갖는 PID를 갖는 패킷은 이 콘텐츠를 제1 비디오 스트림을 갖는 것으로 식별할 수 있고 0x14 값을 갖는 PID를 포함하는 패킷은 이 패킷의 콘텐츠를 제1 오디오 스트림으로 식별할 수 있다. 데이터 랜더마이저(104)는 패킷을 랜더마이징(randomize)하여 이를 리드-솔로몬 (Reed-Solomon) 인코더(106)로 제공한다. 리드-솔로몬 인코더(106)는 20 패러티 바이트를 계산하여 랜더마이징된 데이터에 연결시켜(concatenate) 207 바이트를 갖는 R-S 패킷을 생성한다.

[0007] 컨볼루션 인터리버(108)는 때 맞추어 데이터를 더 랜더마이즈하기 위해 R-S 패킷을 인터리빙한다. 트렐리스 인코더(110)는 인터리빙된 패킷을 인코딩하여 한 블록의 828개의 3비트 심볼을 생성한다. A53 표준은 12개의 트렐리스 인코더의 사용을 규정하는데, 각 트렐리스 인코더는 인터리빙된 패킷에 존재하는 매 2비트에 대해 3비트 심볼을 생성하는 2/3 레이트 트렐리스 인코더이다. 그 결과, 트렐리스 인코더(110)는 디멀티플렉서, 12개의 병렬 2/3 레이트 트렐리스 인코더, 및 멀티플렉서를 포함한다. 컨볼루션 인터리버(108)로부터의 데이터는 디멀티플렉싱되어 12개의 트렐리스 인코더로 분배되고 12개의 트렐리스 인코더에 의해 생성된 심볼들은 심볼들의 스트림이 되도록 멀티플렉싱된다.

[0008] 싱크 멀티플렉서(sync multiplexer)(112)는 각 828 심볼 블록의 시작에서 4개의 미리 정의된 세그먼트 싱크 심볼을 삽입하여 832 심볼 세그먼트를 생성한다. 또한, 싱크 멀티플렉서(112)는 생성되는 매 312개의 세그먼트마다 832개의 심볼을 포함하는 필드 싱크를 삽입한다. 특히, 필드 싱크 심볼들은 312개의 세그먼트보다 선행한다.

[0009] 8-VSB 변조기(114)는 8-VSB (잔류 측파대)를 이용하여 캐리어 신호를 변조하기 위해 트렐리스 인코더(110)에 의해 인코딩된 데이터, 세그먼트 싱크 심볼, 및 필드 싱크를 포함하는 멀티플렉싱된 심볼을 사용한다. 구체적으로, 8-VSB 변조기(114)는 신호를 생성하는데, 이 신호의 진폭은 8개의 이산 레벨 중 하나에 있고, 각 이산 레벨은 특정 3비트 심볼에 대응한다. 이 신호는 그후 여기 도시되지 않은 회로를 이용하여 디지털에서 아날로그 신호 포맷으로 변환되어 무선 주파수로 상향변환(up-converted)된다. 무선 주파수 신호는 안테나(116)를 이용하여 전송된다. 일반적으로, 데이터 랜더마이저(104), 리드-솔로몬 인코더(106), 컨볼루션 인터리버(108) 및 트렐리스 인코더(110)의 조합을 8-VSB 인코더(120)라 한다. 8-VSB 인코더(120)는 A53 인코더 또는 ATSC 인코더라고도 할 수 있다.

[0010] 전송 소스(102)에 의해 생성된 데이터는 ISO/IEC (International Standards Organization/International Electrotechnical Commission) 13818-2 포맷과 등가인 MPEG (motion picture entertainment group) 2 포맷을 이용하여 인코딩되는 소스인 비디오를 포함한다. 전송 소스(102)는 또한 AC-3 (Dolby Arc Consistency algorithm #3)을 이용하여 인코딩되는 소스인 오디오 데이터를 포함한다. A53 표준은 또한 프로그램 가이드 데이터와 같은 다른 프로그램 요소에 대한 메타데이터의 사용을 허용하고 이러한 프로그램 요소는 다른 방법을 이용하여 인코딩되는 소스일 수 있다. 또한, A53 표준은 표준 화질 인터레이스 텔레비전 품질부터 프로그레시브 스캔 와이드스크린 고선명 화질에 이르는 다양한 디스플레이 포맷 및 비디오 품질 레벨에서의 비디오의 전송을 허용한다. FCC는 방송사가 A53 표준을 이용하여 전송 소스(102)에 의해 생성된 데이터를 인코딩할 것을 요구한다. 디지털 텔레비전 프로그램 방송의 전송이 할당된 채널의 전체 19Mb/초 용량을 필요로 하지 않는다면, 방송하는 자는 다른 서비스들을 방송하는 데에, 가능하다면 심지어 휴대용 수신기 및 셀룰라 폰과 같은 장치에게 방송하는 데에 여분의 용량을 활용할 수 있다. 그러나, FCC는 여분의 용량을 사용하여 이러한 장치에 전송되는 데이터는 A53 표준에 따라 전송될 것을 요구한다. A53 표준의 개정이 가능하고 ATSC에 의해 고려되고 있지만, 기존의 또는 소위 레거시(legacy) 디지털 텔레비전 수신기가 계속 사용될 수 있도록 개발이 이뤄져야 한다. 유사하게는, 기존의 A53 표준에 따른 신호의 인코딩 및 전송을 레거시 인코딩 및 전송이라고도 할 수 있다.

[0011] 도 2는 기존 또는 레거시 A53 표준에 부합하는 수신 신호로부터 소스 정보를 추출하기 위해 사용될 수 있는 수

신기(200)의 블록도이다. 안테나(202)는 공중파(airwaves)를 통해 전송되는 전자기(electromagnetic) 신호로부터 수신 전기 신호를 디벨롭(develop)한다. 아날로그-디지털 (A/D) 컨버터(204)는 수신 신호의 디지털 샘플을 생성하고 트렐리스 디코더(206)는 디지털 샘플을 디코딩하여 트렐리스 디코딩된 비트 추정치(estimates)의 스트림을 데이터 스트림으로 생성한다. A/D 컨버터(204)는 또한 수신 신호 내에서 원하는 채널을 수신하기 위한 튜너와 같은 추가 프론트 엔드 처리 회로를 포함할 수 있다. 기존 또는 레거시 A53 표준에 따르면, 트렐리스 디코더(206)는 신호 디멀티플렉서, 12개의 2/3 레이트 트렐리스 디코더 및 신호 멀티플렉서를 포함한다. 디멀티플렉서는 12개의 2/3 레이트 트렐리스 디코더 사이에서 디지털 샘플을 분배하고 멀티플렉서는 12개의 2/3 레이트 트렐리스 디코더 각각에 의해 생성된 추정치를 멀티플렉싱한다.

[0012] 컨볼루션 디인터리버(208)는 트렐리스 디코딩된 비트 추정치의 스트림을 디인터리빙하여 207 바이트를 포함하도록 배열된 시퀀스 또는 패킷을 생성한다. 패킷 배열은 동기 신호들의 위치의 결정 및 식별과 연관하여 수행된다 (도시되지 않음). 리드-솔로몬 오류 정정 회로(210)는 디인터리버(208)에 의해 생성된 207 바이트의 각 시퀀스를 하나 또는 그 이상의 코드워드로 간주하고 코드워드나 패킷에서의 임의의 바이트가 전송 중 오류에 의해 잘못되었는지 결정한다. 이 결정은 코드워드들에 대한 오류 패턴들이나 신드롬(syndromes) 집합을 계산하고 평가함으로써 수행된다. 오류가 검출되면, 리드-솔로몬 오류 정정 회로(210)는 패러티 바이트에서 인코딩된 정보를 이용하여 오류 바이트 복구를 시도한다. 결과적인 오류 정정된 데이터 스트림은 디랜더마이저 (de-randomizer)(212)에 의해 디랜더마이즈된 후 전송되는 콘텐츠의 종류에 따라 데이터 스트림을 디코딩하는 데이터 디코더(214)에 제공된다. 일반적으로, 트렐리스 디코더(206), 디인터리버(208), 리드-솔로몬 디코더(210), 및 디랜더마이저(212)의 조합은 수신기(200) 내에서 8-VSB 디코더(220)로 식별된다. 일반적으로 레거시 A53 표준에 부합하는 신호를 수신하기 위한 일반적인 수신기는 전송 프로세스의 역순으로 수신 프로세스를 수행함에 유의하는 것이 중요하다.

[0013] 일반적으로, 리드-솔로몬 인코딩 및 디코딩에서 이용되는 알고리즘은 당업자에게 잘 알려져 있다. 상기한 바와 같이, 도 1의 리드-솔로몬 인코더(106)는 20 패러티 바이트를 187 바이트를 갖는 데이터 패킷에 부가함으로서 207 바이트를 갖는 코드워드를 생성한다. 도 2의 리드-솔로몬 디코더(210)는 10 바이트까지의 코드워드 내에서 오류를 정정하기 위해 인코더에 의해 부가된 20 바이트를 이용한다.

[0014] 리드-솔로몬 오류 정정 알고리즘은 갈루아 필드 (Galois Field)의 특성을 이용한다. 구체적으로, 갈루아 필드 $GF(p^n)$은 유한 수의 성분 p^n (p 및 n은 정수)을 포함하는 수학적 집합이다. 특정 갈루아 필드는 생성 다항식 g(x)를 이용하여 정의된다. 갈루아 필드의 각 성분은 n 비트를 갖는 유일 비트 패턴으로 표현될 수 있다. 또한, p^n 차 유일 다항식은 각 성분과 연관될 수 있고 여기서 다항식의 각 계수는 0과 p-1 사이의 값이다. 또한, 갈루아 필드에서의 수학적 연산은 중요한 성질을 갖고 있다. 갈루아 필드 $GF(p^n)$의 두 성분의 덧셈은 가산되는 두 성분과 연관된 다항식들의 계수들의 모듈로-p 합(mudulo-p sum)인 계수들을 갖는 다항식과 연관된 성분으로 정의된다. 유사하게, 두 성분의 곱셈은 두 성분과 연관된 다항식들과 갈루아 필드와 연관된 생성 다항식 g(x) 간의 모듈로(modulo) 연산값들의 곱셈으로 정의된다(Similarly, multiplication of two elements is defined as the multiplication of the polynomials associated with the two elements modulo the generator polynomial g(x) associated with the Galois Field). 덧셈 및 곱셈 연산자는 갈루아 필드의 임의의 두 성분의 합 및 곱이 갈루아 필드의 성분이 되도록 갈루아 필드에 대해 정의된다. 리드-솔로몬 코드워드의 특성은, 코드워드의 각 바이트를 갈루아 필드의 한 성분으로 곱하는 것이 결과적으로 다른 유효한 리드-솔로몬 코드워드로 된다는 점이다. 또한, 두개의 리드-솔로몬 코드워드의 바이트끼리의(byte-by-byte) 덧셈은 또다른 리드-솔로몬 코드워드를 생성한다. 레거시 A53 표준은 리드-솔로몬 알고리즘에서 사용을 위한 256 성분 갈루아 필드 $GF(2^8)$ 및 관련된 생성 다항식 g(x)을 정의한다. 갈루아 필드의 특성은 또한 오류를 결정하기 위해 코드워드에 대한 신드롬을 생성하는 능력을 생성한다.

[0015] 디랜더마이저로부터의 출력 패킷은 데이터 디코더(214)에 제공된다. 데이터 디코더(214)는 패킷으로 반송되는 정보의 종류와 이러한 정보를 디코딩하는 방법을 결정하기 위해 디코딩된 패킷의 헤더에서 PID를 이용한다. 헤더에 있는 PID는, 데이터 스트림의 일부로서 주기적으로 전송되고 수신기에서 주기적으로 업데이트될 수 있는 프로그램 맵 테이블 (PMT)에 있는 정보와 비교된다. 데이터 디코더(214)는 인식되는 종류가 아닌 데이터 패킷에 대한 PID를 갖는 임의의 패킷을 무시한다. 이런 식으로, 레거시 A53 표준은 전송 소스가 새로운 패킷 종류에 대해 고유한 PID 값을 할당할 수 있게 함으로써 원래 표준에서는 고려되지 않은 새로운 패킷 종류의 생성을 허용한다. 새로운 패킷 종류를 지원하지 않는 레거시 디코더는 이러한 패킷을 무시할 것이고 새로운 패킷 종류

를 인식하는 새로운 디코더는 이러한 패킷을 처리할 수 있다.

[0016] 명백한 바와 같이, 수신기(200)에서 2/3 레이트 트렐리스 디코더(206) 및 리드-솔로몬 디코더(210)에 의해 적절히 디코딩되는 패킷들만이 데이터 디코더(214)로 제공될 것이다. 트렐리스 디코더(206) 또는 리드-솔로몬 디코더(210)가 패킷을 디코딩할 수 없으면, 수신기는 일반적으로 이러한 패킷을 오류 패킷으로 취급하여 폐기한다. 너무 많은 오류 패킷이 수신되면, A53 표준에 따르는 신호를 수신할 수 있는 일부 수신기는 송신기와의 재동기를 시도할 수 있다.

[0017] A53 표준에 따르는 신호는 일반적으로 동축 케이블이나 전화 회선을 통한 전송을 포함하여 공중(air)이 아닌 다른 방식으로 전송될 수 있다는 것에 유의한다.

[0018] 기존 또는 레거시 A53 표준은 현재 일반적으로 (예컨대, 집에서와 같이) 고정되어 있고 전송된 신호를 캡처하기 위한 대형 안테나에 결합된 수신기에 의한 의도된 사용을 위해 신호를 생성하고 전송하는 것을 정의한다. 그러나, 전송 신호는 휴대용 텔레비전, 차량용 텔레비전, 셀룰라 폰, PDA 등에서 사용되는 작은 안테나를 갖는 수신기나 모바일 수신기가 이러한 신호에서 인코딩된 소스 데이터를 효과적으로 추출할 수 있게 할 정도로 충분히 튼튼하거나(rugged) 강건(robust)하지 않다. 특히, 2/3 레이트 트렐리스 인코더에 의해 제공되는 리던던시(redundancy)는 충분하지 않고, 더 낮은 레이트의 인코더(즉, 더 큰 리던던시를 갖는 인코더)가 모바일 애플리케이션에 필요하다. 따라서, 이동용, 휴대용 및 보행자용 장치에서 향상된 수신기를 이용하여 더 잘 수행하도록 적용된 더 강건한 인코딩 프로세스를 도입하는 것이 바람직하다.

발명의 상세한 설명

[0019] 본 실시예의 일 양태에 따르면, 데이터 페이로드 및 헤더를 갖는 데이터의 패킷을 수신하는 단계, 패킷 내의 데이터를 바이트 코드 인코딩하는 단계 및 바이트 코드 인코딩에 응답하여 헤더 내의 정보를 변경하는 단계를 포함하는 방법이 설명된다.

[0020] 본 실시예의 또 다른 양태에 따르면, 데이터 페이로드 및 헤더를 갖는 데이터의 패킷을 수신하고 데이터의 패킷 내의 데이터를 바이트 코드 인코딩 프로세스를 사용하여 데이터를 인코딩하기 위한 인코더, 및 인코더 블록에 결합되고 인코딩된 데이터의 패킷을 수신하고 바이트 코드 인코딩에 응답하여 헤더 내의 정보를 변경하기 위한 헤더 수정기를 포함하는 장치가 설명된다.

[0021] 본 실시예의 또 다른 양태에 따르면, 복수의 데이터 패킷을 수신하기 위한 패킷 식별자 - 데이터 패킷 각각은 데이터 페이로드 및 헤더를 갖고, 패킷 식별자는 헤더 내의 정보에 기초하여 복수의 패킷으로부터 데이터 패킷을 식별함 -, 패킷 식별자에 결합되고, 복수의 데이터 패킷을 수신하고 바이트 코드 디코딩 프로세스를 사용하여 상기 식별된 데이터 패킷을 디코딩하기 위한 바이트 코드 디코더, 및 바이트 코드 디코더에 결합되고, 리드-솔로몬 디코딩 프로세스를 사용하여 적어도 상기 디코딩된 식별된 데이터 패킷을 디코딩하기 위한 리드-솔로몬 디코더를 포함하는 장치가 설명된다.

[0022] 본 실시예의 또 다른 양태에 따르면, 각각 데이터 페이로드 및 헤더를 갖는 복수의 데이터 패킷을 수신하는 단계, 헤더 내의 정보에 기초하여 상기 복수의 패킷으로부터 데이터 패킷을 식별하는 단계, 식별된 데이터 패킷을 바이트 코드 디코딩 프로세스를 사용하여 디코딩하는 단계, 및 적어도 상기 디코딩된 식별된 데이터 패킷을 리드-솔로몬 디코딩 프로세스를 사용하여 디코딩하는 단계를 포함하는 방법이 설명된다.

실 시 예

[0048] 본 발명의 하나 또는 그 이상의 특정 실시예에 대해 이하에서 설명하기로 한다. 이 실시예의 간명한 설명을 제공하기 위해, 실제 구현예의 모든 특징이 본 명세서에서 설명되지는 않았다. 임의의 이러한 실제 구현예의 전개에서, 임의의 엔지니어링 또는 설계 프로젝트에서와 같이, 다수의 구현예 특정적인 결정은 구현예마다 다를 수 있는 시스템 관련 및 비즈니스 관련 제약의 준수와 같은 개발자의 특정 목적을 달성하기 위해 이루어져야 한다. 또한, 이러한 개발 노력은 본 발명의 혜택을 입은 통상의 기술자에게는 설계, 제조 및 제작하는데 있어서 일상적인 과제임은 물론이다.

[0049] 이하의 설명은 텔레비전 방송 신호에 관한 시스템을 기술하는데, 예를 들어 특히 미국에서의 사용을 위해 정의된 방송 신호에 관한 시스템을 기술한다. 본 실시예들은 이동용, 휴대용 또는 보행용 장치에서 사용될 수 있다. 이러한 장치의 예로는 셀룰라 폰, 인텔리전트 폰, PDA, 랩탑 컴퓨터 및 휴대용 텔레비전이 있지만, 이에 한정되는 것은 아니다. 다른 종류의 신호를 송수신하기 위해 이용되는 다른 시스템은 유사한 구조 및 프로세스

를 포함할 수 있다. 당업자라면 여기 설명된 회로 및 프로세스의 실시예는 가능한 실시예들의 일 집합일 뿐이라는 것을 이해할 것이다. 이로써, 다른 실시예에서, 시스템의 컴포넌트들이 재배열되거나 생략될 수 있고, 또는 다른 컴포넌트들이 추가될 수 있다. 예컨대, 사소한 수정으로, 본 시스템은 세상의 다른 장소에서 사용되는 서비스를 포함하여 위성 비디오 및 오디오 서비스 또는 전화 데이터 서비스에서의 사용을 위해 구성될 수 있다.

[0050] 도 3을 참조하면, 인코더(300)의 일 실시예의 블록도가 도시되어 있다. 인코더(300)는 데이터를 인코딩하여 튼튼하거나 강건한 (rugged or robust) 데이터 스트림을 낳도록 적응된 처리 회로를 포함한다. 러기드(rugged) 데이터 스트림이 되도록 데이터를 인코딩하는 것은 어려운 전송 환경 하에서 데이터의 복구를 허용한다. 예컨대, 인코더(300)에 의해 생성된 러기드 데이터 스트림은 휴대용, 이동용 또는 보행자용 수신 장치에 의한 방송 텔레비전 신호의 개선된 수신을 허용한다. 전송 소스(302)는 랜더마이저(304)에 연결된다. 랜더마이저(304)는 신호 M을 리드-솔로몬 인코더(306)에 제공하고 이에 연결된다. 리드-솔로몬 인코더(306)는 신호 C를 바이트 코드 인코더(308)에 제공하고 이에 연결된다. 바이트 코드 인코더(308)는 두개의 신호 A 및 A'를 인터리버(310)에 제공하고 이에 연결된다. 인터리버(310)는 출력 신호 I를 트렐리스 인코더(312)에 제공하고 이에 연결된다. 특정 블록들은 이전에 설명된 블록과 유사하고 따라서 여기서 상술되지 않는다.

[0051] 도 3에서 데이터 랜더마이저(304)는 전송 소스(302)로부터의 데이터 패킷의 스트림을 랜더마이즈한다. 데이터 패킷 스트림은 187 바이트의 그룹들이 되도록 조직된다. 데이터 패킷 스트림에 대한 다른 배열도 가능함에 유의한다. 또한, 각 데이터 패킷은 하나 또는 그 이상의 코드워드를 포함할 수 있다. 리드-솔로몬 인코더(306)는 각각의 187 바이트 랜더마이즈된 패킷을 인코딩하여 하나 또는 그 이상의 코드워드를 포함하는 207 바이트 패킷을 생성한다. 리드-솔로몬 인코더(306)는 일반적으로 새로운 20 바이트를 생성하고 이 20 바이트를 187 바이트 코드워드의 끝에 첨부한다. 바이트 코드 인코더(308)는 신호 C를 더 수신하고 각각의 207 바이트 리드-솔로몬 패킷을 인코딩하여 추가 207 바이트 코드워드를 생성한다. 일 실시예에서, 바이트 코드 인코더(308)는 레이트 1/2 인코더이다. 레이트 1/2 인코더는 입력에서 제공된 각각의 코드워드에 대해 2개의 출력 코드워드를 제공한다. 각각의 코드워드는 도 3에 도시된 바와 같이 신호 A 및 A'처럼 별개로 제공될 수 있다. 다른 코드 레이트 인코더가 바이트 코드 인코더(308)에 대해 사용될 수 있고 이하에서 상술하기로 한다. 컨볼루션 인터리버(310)는 207 바이트 코드워드의 각각을 인터리빙하고 그 결과를 신호 I로 변조 및 전송에 대비하여 트렐리스 인코더(312)로 제공한다. 리드-솔로몬 인코더(306), 컨볼루션 인터리버(310), 및 트렐리스 인코더(312)는 레거시 A53 표준과 같은 기존 텔레비전 방송 표준에 따르는 레거시 송신기에서 사용되는 것과 동일할 수 있다.

[0052] 상기한 바와 같이, 레이트 1/2 바이트 코드 인코더(308)에 의해 생성된 두개의 코드워드나 패킷은 원래 입력된 리드-솔로몬 패킷의 사본(duplicate)과 리드-솔로몬 패킷에 리던던시를 제공하는 새로운 코드워드를 포함한다. 두개의 코드워드는 또한 체계적 데이터 및 비체계적 데이터로 설명될 수 있다. 체계적 및 비체계적 데이터를 나타내는 코드워드는 더 큰 데이터 구조를 형성하도록 배열될 수 있음에 유의한다. 바람직한 실시예에서, 코드워드는 데이터 패킷의 러기드 데이터 스트림을 형성하도록 조직될 수 있다. 러기드 데이터 스트림은 스트림 A에서 데이터 패킷의 사본인 체계적 패킷과 스트림 A'에서 바이트 코드 인코더의 처리에 의해 생성된 비체계적 패킷을 포함한다. 비체계적 패킷은 또한 러기드 데이터 스트림의 다른 체계적 및 비체계적 패킷으로부터 도출될 수 있는 패킷을 포함한다. 또한, 러기드 데이터 스트림내 패킷은 체계적 바이트 및 비체계적 바이트로 더 구성될 수 있다. 이러한 실시예에서, 체계적 바이트는 콘텐츠 데이터의 바이트의 사본이고 비체계적 바이트는 다른 체계적 및 비체계적 바이트로부터 도출된 것이다.

[0053] 바이트 코드 인코더에 의해 출력된, 리던던트 또는 비체계적 코드워드나 패킷은 리드-솔로몬 패킷의 각 요소(element)를 갈루아 필드의 성분 b에 곱한 결과이다. 일 실시예에서, 전송 소스(302) 및 데이터 랜더마이저(304)가 바이트 M(1), M(2),..., M(187) (여기서, M(1)은 메시지의 제1 바이트, M(2)는 메시지의 제2 바이트 등)로 구성된 메시지 M을 생성하면, 리드-솔로몬 인코더(306)는 메시지 M으로부터 패킷 또는 코드워드 C를 생성한다 (여기서, 코드워드 C는 바이트 C(1), C(2),..., C(207) 포함). 다음에, 바이트 코드 인코더(308)는 코드워드 C로부터 코드워드 A 및 A'를 다음과 같이 생성한다.

$$A(i) = C(i) \quad i=1,2, \ldots,207 \qquad (1)$$

[0054]

$$A'(i) = b * C(i) \quad i=1,2, \ldots,207 \qquad (2)$$

[0055] 값 b는 리드-솔로몬 인코더(306)에 의해 사용될 수 있는 동일한 갈루아 필드 $GF(2^8)$의 미리 결정된 (영이 아닌 (non-zero)) 성분이다. 일 실시예에서, b 성분의 값은 2이다. 코드워드 A'도 리드-솔로몬 코드워드임은 명백하다. 즉, 레거시 A53 표준에 따르는 수신기에서 리드-솔로몬 디코더는 신호 전송 중에 코드워드 A'에 오류가

도입되었는지를 결정하기 위해 코드워드 A'로부터 유효한 신드롬들을 계산하고 계산된 신드롬들을 이용하여 임의의 이러한 오류를 정정한다.

[0056] 도 4를 참조하면, 인코더(400)의 다른 실시예의 블록도가 도시되어 있다. 인코더(400)에서 블록은 기본적으로 인코더(300)에 대해 이전에 설명된 것과 동일한 기능을 갖는다. 그러나, 일부 블록은 동작 순서로 재배열되었다. 바이트 코드 인코더(406)가 리드-솔로몬 인코더(408) 앞에 위치하였다. 이 배열에서, 전송 소스(402) 및 데이터 랜더마이저(404)는 메시지 스트림 M을 생성한다. 이 메시지 스트림은 예컨대 187 바이트 패킷으로 배열될 수 있다. 1/2 레이트 인코더로서 도시된 바이트 코드 인코더(406)는 각각의 들어오는(incoming) 187 바이트 메시지 패킷에 대해 두개의 187 바이트 패킷을 생성한다. 이전처럼, 두개의 187 바이트 패킷은, A로 표기되어 체계적 패킷으로 식별된 메시지 패킷의 사본과 함께, 리던던트 데이터를 포함하는 비체계적 패킷으로 식별된 새로운 패킷 A'를 포함한다. 도 4에서 A 및 A'로 표기된 패킷은 도 3에서 패킷 A 및 A'과 다르다는 것에 유의해야 한다.

[0057] 리드-솔로몬 인코더(408)는 두개의 패킷 A 및 A'를 차례로 인코딩하여 두개의 207 바이트 코드워드 C 및 C'를 각각 생성한다. 컨볼루션 인터리버(410)는 각각의 코드워드 C 및 C'를 인터리빙하고 트렐리스 인코더(412)는 그후 변조를 대비하여 인터리빙된 데이터를 인코딩한다. 바이트 코드 인코더(406) 및 리드-솔로몬 인코더(408)에 의해 사용되는 갈루아 필드 산술의 특성은 코드워드 C 및 C'가 코드워드 A 및 A'와 각각 동일하다는 것을 보증한다. 블록들의 재순서화가 블록 처리때문에 가능하고 기본적인 수학적 관계가 동일 갈루아 필드에 대한 선형 연산임에 유의해야 한다.

[0058] 도 5를 참조하면, 인코더(500)의 다른 실시예의 블록도가 도시되어 있다. 이전처럼, 인코더(500)에서의 블록들은 기본적으로 인코더(300 및 400)에 대해 이전에 설명된 것과 동일한 기능을 갖는다. 그러나, 일부 블록은 동작 순서로 재배열되었다. 바이트 코드 인코더(504)가 랜더마이저(506) 전에 위치하였다. 데이터 생성기(502)는 예컨대 187 바이트 데이터 패킷으로 배열된 메시지 스트림을 생성한다. 각각의 이러한 데이터 패킷에 대해 바이트 코드 인코더(504)는 체계적(systematic) 패킷(즉, 데이터 패킷의 사본)을 생성하고 추가적인 비체계적(non-systematic) 패킷(즉, 데이터 패킷의 바이트 코드 인코딩에 의한 바이트 인코딩된 패킷)을 생성한다. 바이트 코드 인코더(504)에 의해 생성된 패킷들(원래 데이터 및 인코딩된 것)은 레거시 8-VSB 신호 인코더(530)에게 제공될 수 있는 러기드 데이터 스트림에 포함된다. 레거시 8-VSB 신호 인코더(530)는 도 1에 도시된 8-VSB 인코더(120)와 기능이 유사하다. 레거시 8-VSB 인코더(530)는 데이터 랜더마이저(506), 리드-솔로몬 인코더(508), 컨볼루션 인터리버(510) 및 트렐리스 인코더(512)를 포함한다. 레거시 8-VSB 인코더(530)는 앞서 설명된 바와 같이 A53 표준에 따르는 체계적 패킷 및 비체계적 패킷을 동일하게 인코딩한다. 도시된 바와 같이, 데이터 생성기(502) 및 바이트 코드 인코더(504)는 전송 소스(520)의 일부로서 간주될 수 있다. 또한, 러기드 또는 로버스트 데이터 스트림을 생성하는 바이트 코드 인코더의 추가 기능이 레거시 8-VSB 인코더(530)와 같은 기존 전송 설비의 기존 하드웨어 구조에 대한 최소한의 변경으로 추가될 수 있다. 랜더마이저(506) 및 바이트 코드 인코더(504)의 재위치 지정이 블록 처리때문에 가능하고 기본적인 수학적 관계가 동일 갈루아 필드에 대한 선형 연산이라는 점에 유의해야 한다. 리드-솔로몬 인코더와 같이 랜더마이저(506)는 데이터내 각각의 바이트를 일정한 값으로 곱하는 선형 연산을 이용한다.

[0059] 바이트 코드 인코더(504)는 PID를 포함한 헤더를 형성하는 바이트를 포함하여 데이터 패킷의 모든 바이트를 인코딩하여 러기드 데이터 스트림의 하나 또는 그 이상의 비체계적 패킷을 생성한다. 따라서, 각 비체계적 패킷의 PID는 바이트 코드 인코딩되고, 수신 장치에 의해 인식가능한 PID 값을 더 이상 나타낼 수 없다.

[0060] 인코더(500)로 묘사된 송신기 실시예에 의해 인코딩된 임의의 패킷은 도 2에서 설명된 디코더(220)와 유사한 디코더의 일 실시예에 의해 디코딩될 수 있음은 물론이다 (예컨대, A53 표준에 따르는 레거시 수신기). 디코더(216)는 러기드 데이터 스트림의 패킷을 데이터 디코더(214)로 제공한다. 러기드 데이터 스트림은 바이트 코드 인코더를 이용하여 인코딩되었고 디코더(220)에 의해서는 올바로 디코딩되지 않을 비체계적 패킷을 포함한다. 그러나, 이러한 패킷은 기존 또는 레거시 데이터 포맷으로 프로그램 맵 테이블(PMT)과 연관되지 않은 PID를 갖기 때문에, 레거시 수신기내 데이터 디코더(214)는 러기드 데이터 스트림의 이런 비체계적 패킷을 무시한다.

[0061] 바이트 코드 인코더(504)는 상기 수학식 (2)를 이용하여 각각의 체계적 패킷에 대한 비체계적 패킷을 생성하고 양 패킷의 전송을 위해 레거시 8-VSB 인코더(530)로 제공하는데, 유효 데이터 레이트 1/2 (즉, 1바이트 인, 2바이트 아웃)을 갖는 인코딩된 스트림을 생성한다. 상기한 바와 같이, 바이트 코드 인코더(504)는 다른 유효 데이터 레이트를 생성하기 위해 다른 인코딩 레이트를 이용할 수 있다. 일부 실시예에서, 바이트 코드 인코더는 데이터 생성기(502)에 의해 생성된 매 두개의 소스 패킷에 대해 하나의 바이트 인코딩된 패킷을 생성하여 다

음과 같이 계산되는 두 개의 체계적 패킷과 하나의 비체계적 패킷을 포함하는 레이트 2/3의 러기드 데이터 스트림을 생성한다.

[0062] $$M_{AB}(i) = M_A(i)*b_1 + M_B(i)*b_2 \quad i=1,2,\ldots,187 \qquad (3)$$

[0063] 여기서, M_A 및 M_B는 데이터 생성기(502)에 의해 생성된 연속하는 체계적 패킷이고 b_1 및 b_2는 리드-솔로몬 인코더(508)에 의해 사용되는 갈루아 필드와 같은 갈루아 필드의 미리 정해진 성분들이다. 일 실시예에서, b_1 및 b_2 성분의 값은 2이다. 일부 실시예에서, b_1 및 b_2의 값은 동일하지 않을 수 있다. 바이트 코드 인코더(504)는 패킷 M_A, M_B 및 M_{AB}를 추후 인코딩 및 전송을 위해 레거시 8-VSB 인코더(530)로 제공한다.

[0064] 바이트 코드 인코더(504)는 리던던트 패킷을 생성하기 위한 추가 입력 데이터 패킷을 포함시킴으로써 러기드 데이터 스트림(즉, 더 낮은 데이터 레이트를 갖는 것)을 생성하도록 서로 다른 코딩 레이트를 이용할 수 있다. 바이트 코드 인코더(504)의 일 실시예는 데이터 생성기(502)로부터의 4개의 체계적 패킷 M_A, M_B, M_C 및 M_D와 다음과 같이 계산되는 5개의 비체계적 패킷을 이용하여 레이트 4/9 데이터 스트림을 생성한다.

$$M_{AB}(i) = M_A(i)*b_1 + M_B(i)*b_2 \quad i=1,2,\ldots,187 \qquad (4)$$

$$M_{CD}(i) = M_C(i)*b_3 + M_D(i)*b_4 \quad i=1,2,\ldots,187 \qquad (5)$$

$$M_{AC}(i) = M_A(i)*b_5 + M_C(i)*b_6 \quad i=1,2,\ldots,187 \qquad (6)$$

$$M_{BD}(i) = M_B(i)*b_7 + M_D(i)*b_8 \quad i=1,2,\ldots,187 \qquad (7)$$

$$M_{ABCD}(i) = M_{AB}(i)*b_9 + M_{CD}(i)*b_{10} \quad i=1,2,\ldots,187 \qquad (8)$$

[0065]

[0066] 값 b_1, b_2,..., b_{10}은 갈루아 필드로부터 선택된 미리결정된 성분들이다. 일 실시예에서, b_1, b_2,..., b_{10}에 대한 값은 2이다. 또한, 식(8)에서 나타난 바와 같이, 패킷 M_{ABCD}는 다른 리던던트 패킷들, 특히 패킷 M_{AB} 및 M_{CD}로부터만 생성된 리던던트 패킷이다. 리던던트 패킷 M_{ABCD}는 이와 달리 리던던트 패킷 M_{AC} 및 M_{BC}의 요소들을 이용하여 생성될 수 있음은 물론이다. 전송 소스 생성기(520)의 일부 실시예에서, 하나 또는 그 이상의 비체계적 패킷의 제거는 펑츄어링(puncturing)으로 알려진 동작에서 수행될 수 있다. 예컨대, 펑츄어링된 레이트 4/8는 예컨대 리던던트 패킷만을 채택했을 패킷들 중 하나(즉, 이 경우, M_{ABCD})를 생성하지 않음으로써 산출될 수 있다. 코드 펑츄어링은 전송된 패킷이나 코드워드의 수에 대한 특정 제한을 충족시키기 위해 전송 패킷의 수를 변경하도록 사용될 수 있다.

[0067] 또한, 바이트 코드 인코더(504)는 다음과 같이 8개의 데이터 패킷 M_A, M_B,..., M_H을 이용하여 19개의 비체계적 패킷을 생성함으로써 데이터 레이트 8/27을 갖는 러기드 데이터 스트림을 생성할 수 있다.

$$M_{AB}(i) = M_A(i)*b_1 + M_B(i)*b_2 \quad i=1,2,\ldots,187 \quad (9)$$

$$M_{CD}(i) = M_C(i)*b_3 + M_D(i)*b_4 \quad i=1,2,\ldots,187 \quad (10)$$

$$M_{AC}(i) = M_A(i)*b_5 + M_C(i)*b_6 \quad i=1,2,\ldots,187 \quad (11)$$

$$M_{BD}(i) = M_B(i)*b_7 + M_D(i)*b_8 \quad i=1,2,\ldots,187 \quad (12)$$

$$M_{ABCD}(i) = M_{AB}(i)*b_9 + M_{CD}(i)*b_{10} \quad i=1,2,\ldots,187 \quad (13)$$

$$M_{EF}(i) = M_E(i)*b_{11} + M_F(i)*b_{12} \quad i=1,2,\ldots,187 \quad (14)$$

$$M_{GH}(i) = M_G(i)*b_{13} + M_H(i)*b_{14} \quad i=1,2,\ldots,187 \quad (15)$$

$$M_{EG}(i) = M_E(i)*b_{15} + M_G(i)*b_{16} \quad i=1,2,\ldots,187 \quad (16)$$

$$M_{FH}(i) = M_F(i)*b_{17} + M_H(i)*b_{18} \quad i=1,2,\ldots,187 \quad (17)$$

$$M_{EFGH}(i) = M_{EF}(i)*b_{19} + M_{GH}(i)*b_{20} \quad i=1,2,\ldots,187 \quad (18)$$

$$M_{AE}(i) = M_A(i)*b_{21} + M_E(i)*b_{22} \quad i=1,2,\ldots,187 \quad (19)$$

$$M_{BF}(i) = M_B(i)*b_{23} + M_F(i)*b_{24} \quad i=1,2,\ldots,187 \quad (20)$$

$$M_{CG}(i) = M_C(i)*b_{25} + M_G(i)*b_{26} \quad i=1,2,\ldots,187 \quad (21)$$

$$M_{DH}(i) = M_D(i)*b_{27} + M_H(i)*b_{28} \quad i=1,2,\ldots,187 \quad (22)$$

$$M_{ACEG}(i) = M_{AC}(i)*b_{29} + M_{EG}(i)*b_{30} \quad i=1,2,\ldots,187 \quad (23)$$

$$M_{BDFH}(i) = M_{BD}(i)*b_{31} + M_{FH}(i)*b_{32} \quad i=1,2,\ldots,187 \quad (24)$$

$$M_{ABEF}(i) = M_{AB}(i)*b_{33} + M_{EF}(i)*b_{34} \quad i=1,2,\ldots,187 \quad (25)$$

$$M_{CDGH}(i) = M_{CD}(i)*b_{35} + M_{GH}(i)*b_{36} \quad i=1,2,\ldots,187 \quad (26)$$

$$M_{ABCDEFGH}(i) = M_{ABCD}(i)*b_{37} + M_{EFGH}(i)*b_{38} \quad i=1,2,\ldots,187 \quad (27)$$

[0068]

[0069] 또한, 데이터 레이트 8/26을 갖는 펑츄어링된 코드는 패킷 $M_{ABCDEFGH}$ 또는 리던던트 패킷들로부터만 생성된 그 외의 패킷을 생성하지 않음으로써 바이트 코드 인코더(504)에 의해 산출될 수 있다.

[0070] 상기한 바와 같이, 바이트 코드 인코더는 사용된 코드워드나 패킷의 수 및 단일 인코딩 프로세스를 통해 형성된 코드워드나 패킷의 수에 기초하여 특정의 인코딩 코드 레이트들을 생성하도록 구성될 수 있다. 또한, 더 복잡한 코드 레이트들은 앞서 설명한 코드 레이트 인코더의 특정 배열을 빌딩 블록들 또는 성분을 이루는(constituent) 코드 레이트 인코더들로서 이용하여 구성될 수 있다. 또한, 추가 처리 블록들이 연쇄 바이트 코드 인코더를 형성하기 위해 포함될 수 있다. 예컨대, 연쇄 바이트 코드 인코더는 생성된 데이터 스트림의 러기드 특성(ruggedness)을 개선하기 위한 리던던시 이외에 성분을 이루는 바이트 코드 인코더들 사이에 추가적인 인터리빙 블록들을 사용할 수 있다. 연쇄 바이트 코드 인코더의 다양한 실시예에 대해 이하에서 설명하기로 한다.

[0071] 도 6을 참조하면, 연쇄 바이트 코드 인코더(600)의 일 실시예가 도시되어 있다. 연쇄 바이트 코드 인코더는 패킷들이나 코드워드들을 수신하고 이들을 제1의 2/3 레이트 바이트 인코더(602)로 제공한다. 제1의 2/3 레이트 바이트 코드 인코더(602)의 출력은 인터리버(604)로 제공된다. 인터리버(604)의 출력은 제2의 2/3 레이트 바이트 코드 인코더(606)로 제공된다. 제2의 2/3 레이트 바이트 코드 인코더(606)의 출력은 바이트 펑츄어

(puncture) 블록(608)으로 제공된다. 펑츄어 블록(608)의 출력은 데이터 패킷타이저(610)로 제공된다. 데이터 패킷타이저(610)의 출력은 추가 처리 (예컨대, 도 5에서 설명된 레거시 전송 인코딩)를 위해 제공될 수 있다.

[0072] 제1의 2/3 레이트 바이트 코드 인코더(602)는 12 바이트의 콘텐츠 데이터 패킷을 수신하고 12 바이트로부터 제1 바이트 코드 인코딩된 스트림을 생성한다. 12 바이트 중의 매 두개의 콘텐츠 데이터 바이트 M_A 및 M_B에 대해, 제1 바이트 코드 인코딩된 스트림은 바이트 M_A 및 M_B의 사본과 앞서 설명된 바와 같이 계산되는 리던던트 바이트 M_{AB}를 포함한다. 일부 실시예에서, 콘텐츠 데이터 바이트들 M_A 및 M_B는 데이터 생성기 (예컨대, 도 5의 데이터 생성기(502))에 의해 생성된 하나의 콘텐츠 데이터 패킷의 바이트들이다. 다른 실시예에서, 제1의 2/3 레이트 바이트 코드 인코더(602)는 두개의 서로 다른 콘텐츠 데이터 패킷 A 및 B로부터 각각 콘텐츠 데이터 바이트 M_A 및 M_B를 선택한다. 콘텐츠 데이터의 매 12 바이트에 대해, 18 바이트가 제1 바이트 코드 인코딩된 출력 스트림의 일부로서 출력된다.

[0073] 제1 바이트 코드 인코딩된 스트림은 18 인터리빙된 바이트를 포함하는 인터리빙된 스트림을 생성하기 위해 인터리버(604)에 의해 인터리빙된다. 인터리버(604) 뿐만 아니라 이하의 다른 인터리버는 당해 기술분야에서 잘 알려진 임의의 인터리빙 방법(예컨대, 의사랜덤, 행렬, 코드 최적화 등)을 이용할 수 있다. 또한, 인터리버는 전체 인터리버 데이터 길이를 저장할 수 있는 저장 용량을 갖는 메모리를 포함할 수도 있다. 바람직한 실시예에서, 인터리버(604)는 도 7에 도시된 테이블(700)에서 제공된 대로 출력 바이트들을 배열한다. 테이블(700)은 입력에서 바이트들의 위치를 나타내는 행(710)을 포함한다. 행(720)은 바이트들이 출력에서 판독될 때의 순서를 나타낸다. 인터리빙된 스트림은 제2의 2/3 레이트 바이트 코드 인코더(606)로 제공된다. 제2의 2/3 레이트 바이트 코드 인코더(606)는 인터리빙된 스트림에서 18 인터리빙된 바이트의 그룹들을 인코딩하여 27 바이트의 그룹을 포함하는 제2 바이트 코드 인코딩된 스트림을 생성한다. 상기한 바와 같이, 인터리버에 의해 생성된 매 두개의 바이트 I_A 및 I_B에 대해, 제2의 2/3 레이트 바이트 코딩 스트림은 두개의 바이트 I_A 및 I_B의 사본 및 바이트 I_{AB}를 갖는다. 바이트 I_A는 데이터 생성기 (예컨대, 도 5의 데이터 생성기(502))에 의해 생성된 콘텐츠 데이터의 바이트들 중 하나의 사본일 수 있거나 제1 바이트 코드 인코더(602)에 의해 리던던트나 비체계적 바이트로 디벨롭된 바이트일 수 있음은 물론이다. 유사하게, 바이트 I_B는 콘텐츠 데이터의 한 바이트의 사본이거나 제1 바이트 코드 인코더(602)에 의해 리던던트나 비체계적 바이트로 디벨롭된 바이트일 수 있다.

[0074] 선형 인코더에서 사용되는 인터리버들은 전통적으로 인터리버 길이나 깊이가 매우 길다. 연쇄 바이트 코드 인코더들에서 사용되는 인터리버(604)와 같은 인터리버들은 길이가 짧고 코딩 레이트에 대해 최적화된다. 종래 접근법과 달리, 바이트 코드 인터리버들은 예를 들어 적은 레이턴시(latency)를 강조한다.

[0075] 바이트 펑츄어 블록(608)은 제2 바이트 코드 인코딩된 스트림에서 27 바이트의 그룹으로부터 한 바이트를 제거하여 26 바이트 그룹을 포함하는 펑츄어링된 스트림을 생성한다. 바이트 펑츄어링은 주어진 코딩 구조를 위해 제공되고 전송되는 바이트의 수를 감소시킴으로써 데이터 효율을 개선하기 위해 사용된다. 그러나, 개선된 데이터 효율은 데이터 스트림으로부터 하나 또는 그 이상의 인코딩된 바이트의 부재에 기인하여 수신기내 디코딩 회로에서의 결과적인 성능 저하와 상충 관계를 갖는다. 바이트 펑츄어링은 전송 포맷을 위해 편리한 인코딩된 데이터의 패킷 또는 바이트의 그룹 또는 블록을 생성하기 위해 사용될 수도 있다. 특정의 바이트들 또는 패킷들 그룹화에 기초한 코딩 구조들은 종종 블록 코드들이라 부른다.

[0076] 바이트 펑츄어 블록(608)은 제2 인코딩된 스트림으로부터 2 바이트 이상을 제거할 수 있다. 예컨대, 12/24 레이트 데이터 스트림을 생성하기 위해 제거될 수 있는 3 바이트를 식별하는 것이 가능할 수 있다. 1 바이트 이상을 펑츄어링하면 코딩 레이트의 개선을 가져오는 반면 인코딩의 유효성을 더 저하시킬 것이다. 바이트 펑츄어 블록(608)에서의 추가 바이트들의 제거는 인터리버(604)에서의 최적의 인터리빙에 기초하여 달성된다. 이런 식으로 펑츄어링 및 인터리빙은 출력 패킷들의 주어진 출력 블록 크기를 생성하는 것에 기초한 최적의 코드 레이트를 허용하도록 상호작용한다.

[0077] 패킷타이저(610)는 펑츄어링된 스트림으로부터 바이트를 조합하여 187 바이트의 분리된 패킷들이 되도록 그룹화한다. 바이트 코드 인코더(600)의 컴포넌트에 의해 생성된 러기드 데이터 스트림은 12/26 레이트 데이터 스트림을 생성한다. 바이트 코드 인코더(600)는 바이트 펑츄어 블록(608)이 사용되지 않으면 12/27 레이트 데이터 스트림을 생성할 수 있다.

[0078] 연쇄 바이트 코드 인코더(600)와 유사한 연쇄 바이트 코드 인코더들은 상술한 12/27 레이트 및 12/26 레이트의 러기드 데이터 스트림이 아닌 다른 러기드 데이터 스트림을 생성하기 위해 사용될 수 있다. 도 8을 참조하면,

연쇄 바이트 코드 인코더(800)의 일 실시예의 다른 블록도가 도시되어 있다. 연쇄 바이트 코드 인코더(800)는 제1의 2/3 레이트 바이트 코드 인코더(802)가 매 4 바이트의 콘텐츠 데이터에 대해 6 바이트의 그룹을 포함하는 제1 바이트 코드 인코딩된 데이터 스트림을 생성한다는 점을 제외하고 바이트 코드 인코더(600)와 유사하다. 인터리버(804)는 6 바이트를 인터리빙하고 제2의 2/3 레이트 바이트 코드 인코더(806)는 제공된 매 6 바이트에 대해 9 바이트의 그룹을 포함하는 제2 바이트 코드 인코딩된 데이터 스트림을 생성한다. 인터리버(804)는 두개의 2/3 레이트 바이트 코드 디코더의 연쇄에 대한 가장 작은 가능한 인터리버 길이를 나타낸다. 바이트 펑츄어(808)는 제2의 2/3 레이트 바이트 코드 인코더(806)에 의해 생성된 매 9 바이트에 대해 1 바이트를 제거한다. 바이트 코드 인코더(800)에 의해 생성된 러기드 데이터 스트림은 4/8 레이트 바이트 코드로서 인코딩된다. 바이트 코드 인코더(800)는 바이트 펑츄어(808)가 사용되지 않으면 4/9 레이트 바이트 코드를 생성하기 위해 사용될 수 있다.

[0079] 도 9를 참조하면, 연쇄 바이트 코드 인코더(900)의 또다른 실시예의 블록도가 도시되어 있다. 제1 바이트 코드 인코더(902) 및 인터리버(904)는 바이트 코드 인코더(600)의 그것과 동일하다. 그러나, 제2 바이트 코드 인코더(906)는 1/2 레이트 바이트 코드 인코더이다. 1/2 레이트 바이트 코드 인코더(906)는 인터리빙된 스트림에서 18 인터리빙된 바이트의 그룹들을 인코딩하여 27 바이트의 그룹들을 포함하는 제2 바이트 코드 인코딩된 스트림을 생성한다. 상기한 바와 같이, 인터리버(904)에 의해 생성된 매 1 바이트 I에 대해, 1/2 레이트 바이트 코드 인코딩된 스트림은 바이트 I의 사본 및 비체계적 바이트 I'를 포함한다. 바이트 I는 데이터 생성기 (예컨대, 도 5의 데이터 생성기(502))에 의해 생성된 콘텐츠 데이터 바이트들 중 하나의 사본이거나 제1 바이트 코드 인코더(902)에 의해 리던던트 또는 비체계적 바이트로서 디벨롭된 바이트일 수 있음은 물론이다.

[0080] 바이트 펑츄어 블록(908)은 제2 바이트 코드 인코딩된 스트림에서 36 바이트의 그룹으로부터 1 바이트를 제거하여 35 바이트 그룹을 포함하는 펑츄어링된 스트림을 생성한다. 연쇄 바이트 코드 인코더(900)는 12/35 레이트의 펑츄어링된 러기드 데이터 스트림 또는 레이트 12/36의 펑츄어링되지 않은 데이터 스트림을 생성할 수 있다.

[0081] 도 6, 8 및 9가 두개의 구성 바이트 코드 인코더 및 하나의 인터리버를 이용하는 연쇄 바이트 코더 인코더의 실시예들을 도시하지만, 바이트 코더의 다른 실시예들은 다른 데이터 레이트를 갖는 러기드 데이터 스트림을 생성하기 위해 추가의 구성 바이트 코드 인코더 및 인터리버를 포함할 수 있다. 도 10을 참조하면, 연쇄 바이트 코드 인코더(1000)의 다른 실시예의 블록도가 도시된다. 인코더(1000)는 3개의 구성 바이트 코드 인코더, 2개의 인터리버 및 1개의 펑츄어 블록을 포함한다. 1/2 레이트 바이트 코드 인코더(1002)는 데이터 생성기 (예컨대 데이터 생성기(502))로부터 콘텐츠 데이터 바이트를 수신한다. 1/2 레이트 바이트 코드 인코더는 수신된 콘텐츠 데이터의 매 12 바이트에 대해 제1 바이트 코드 인코딩된 스트림에서 24 바이트의 그룹들을 생성한다.

[0082] 제1 인터리버(1004)는 제1 바이트 코드 인코딩된 스트림에서 24 바이트 그룹들을 인터리빙하고 제1 인터리빙된 스트림에서 24 바이트의 인터리빙된 그룹을 제1의 2/3 레이트 바이트 코드 인코더(1006)로 제공한다. 바람직한 실시예에서, 제1 인터리버(1004)는 도 11에 도시된 테이블(1100)에서 제시된 바와 같이 출력 바이트를 배열한다. 테이블(1100)은 입력에서의 바이트들의 위치를 나타내는 행(1110)을 포함한다. 행(1120)은 출력에서 판독되는 바이트들의 순서를 나타낸다. 제1의 2/3 레이트 바이트 코드 인코더(1006)는 제공된 24 바이트의 각 그룹에 대해 36 바이트의 그룹을 포함하는 제2 바이트 코드 인코딩된 스트림을 생성한다. 제2 인터리버(1008)는 36 바이트의 각 그룹을 18 바이트씩 두 세트로 나눔으로써 36 바이트 제2 바이트 코드 인코딩된 스트림을 인터리빙한다. 제2 인터리버(1008)는 18 바이트의 각 세트를 인터리빙하고 이 인터리빙된 데이터를 두개의 18 바이트 인터리빙된 스트림으로서 제2의 2/3 레이트 바이트 코드 인코더(1010)로 제공한다. 제2의 2/3 레이트 바이트 코드 인코더(1010)는 제1의 2/3 레이트 바이트 코드 인코더(1006)와 유사한 방식으로 동작하고 각 18 바이트 인터리빙된 스트림을 인코딩하여 27 바이트의 그룹들을 포함하는 제3 바이트 인코딩된 스트림을 생성한다. 바이트 펑츄어 블록(1012)은 제3 바이트 인코딩된 스트림의 1 바이트를 펑츄어링하고 26 바이트들을 패킷타이저(1014)로 제공한다. 패킷타이저(1014)는 인터리버(1008)에 의해 분리된 26 바이트의 세트들을 다시 그룹화한다. 패킷타이저(1014)는 또한 상기한 바와 같이 펑츄어링된 스트림으로부터 바이트를 조합하여 이를 187 바이트의 이산 패킷들로 그룹화한다. 제2의 2/3 레이트 바이트 코드 인코더(910)가 인코더(1000)의 입력에서 수신된 매 12 바이트에 대해 2개의 27 바이트 바이트 코드 인코딩된 스트림들을 생성하기 때문에 연쇄 바이트 코드 인코더(1000)는 12/54의 펑츄어링되지 않은 러기드 데이터 스트림이나 12/52의 펑츄어링된 데이터 스트림을 생성한다.

[0083] 도 12를 참조하면, 연쇄 바이트 코드 인코더(1200)의 또다른 실시예의 블록도가 도시되어 있다. 연쇄 바이트 코드 인코더(1200)는 병렬로 연결되어 동작하는 두개의 성분 바이트 코드 인코더를 포함한다. 즉, 이 연쇄는 앞서 설명한 직렬 연쇄와 달리 병렬 연쇄이다. 입력 스트림으로부터의 17 바이트는 16 바이트의 제1 그룹 및 1

바이트의 제2 그룹으로 나뉜다. 2/3 레이트 바이트 코드 인코더(1210)는 16 바이트의 제1 그룹을 수신하고 수신된 매 16 바이트의 콘텐츠 데이터에 대해 24 바이트의 제1 바이트 코드 인코딩된 스트림을 생성한다. 1/2 레이트 바이트 코드 인코더(1220)는 1 바이트의 제2 그룹을 수신하고 수신된 매 1 바이트의 콘텐츠 데이터에 대해 2 바이트의 제2 바이트 코드 인코딩된 스트림을 생성한다. 24 바이트의 그룹들을 포함하는 제1 바이트 코드 인코딩된 스트림 및 2 바이트의 그룹들을 포함하는 제2 바이트 코드 인코딩된 스트림은 연쇄되어 26 바이트의 그룹들을 포함하는 최종 바이트 코드 인코딩된 스트림을 형성한다. 연쇄 바이트 코드 인코더(1200)는 레이트 17/26의 펑츄어링되지 않은 러기드 데이터 스트림을 생성한다.

[0084] 도 6, 8, 9, 10 및 12에 도시된 것과 다른 레이트를 갖는 연쇄 바이트 코드 인코더가 다양한 코드 레이트로 러기드 데이터 스트림을 생성하기 위해 사용될 수 있음은 당업자에게 명백하다. 마찬가지로, 다른 종류나 배열의 인터리버나 펑츄어 블록이 상기 실시예에서 사용된 것 대신에 사용될 수 있다.

[0085] 도 13을 참조하면, 인코더(1300)의 또다른 실시예의 블록도가 도시된다. 인코더(1300)는 도 5에 도시된 인코더(500)의 대안으로서 MPEG 전송 스트림 소스(1302)를 포함한다. MPEG 전송 스트림 소스(1302)는 몇개의 추가 블록을 포함하는 ATSC M/H 블록(1310)에 연결된다. ATSC M/H 블록(1310) 내에 포함된 블록들은 들어오는 데이터 스트림을 처리하고 이동용, 보행자용 및 휴대용 장치에 의한 수신 및 사용을 위해 적응된 러기드 데이터 스트림을 생성한다. 이 블록에 대해서는 추후 더 설명하기로 한다. ATSC M/H 블록(1310)은 역시 내부에 몇개의 추가 블록을 포함하는 ATSC A53 레거시 블록(1350)에 연결된다. ATSC A53 레거시 블록(1350) 내에 포함된 데이터 랜더마이저(1352), 리드-솔로몬 인코더(1354), 컨볼루션 바이트 인터리버(1356), 트렐리스 인코더(1358), 싱크 삽입 블록(1360), 및 변조 블록(1362)은 도 1에서 설명한 블록들과 유사하다. 따라서, 이 블록들에 대해서는 더 설명하지 않는다.

[0086] ATSC M/H 블록(1310) 내에서, 패킷 인터리버(1312)는 패킷들로 배열된 데이터 스트림을 수신한다. 각 패킷은 187 바이트를 포함하고 패킷 식별을 위해 사용되는 3 바이트 헤더를 포함한다. 패킷 인터리버(1312)의 출력은 GF(256) 직렬 연쇄 블록 코더 (SCBC)(1314)로 제공된다. GF(256) SCBC(1314)의 출력은 패킷 디인터리버(1316)에 연결된다. 패킷 디인터리버(1316)의 출력은 전송 스트림 헤더 수정기(1318)에 연결된다. 전송 스트림 헤더 수정기(1318)의 출력은 아프리오리(a-priori) 트랜스포트 패킷 삽입기(1320)에 연결된다. 아프리오리 트랜스포트 패킷 삽입기(1320)의 출력은 ATSC A53 레거시 인코더(1350)에 연결된다.

[0087] 패킷 인터리버(1312)는 행들로 배열된 패킷들로서 수신된 데이터를 패킷들의 행들로부터의 바이트의 열들에 기초한 코드워드들로 재배열한다. 패킷 인터리버(1312)는 도 14에 도시된 바와 같이 행별 순서로 고정된 수의 연속 패킷으로부터 바이트를 취하고, 이 바이트를 도 15에 도시된 바와 같이 열별로 출력한다. 특히, 도 14 및 도 15는 12 행의 187 바이트 패킷들의 판독 및 187 열의 12 바이트 코드워드들의 출력을 도시한다. 패킷 인터리빙의 결과, 바이트 0으로 표기된 모든 제1 바이트들이 함께 그룹화되고, 바이트 1로 표기된 모든 제2 바이트들이 함께 그룹화되고, 그 이후에 대해서도 마찬가지로 그룹화된다. 인터리버 내로 판독된 패킷들의 수는 소스 프레임을 구성하고 GF(256) SCBC(1314)에서의 처리를 위해 필요한 소스 코드워드들이나 심볼들의 수와 동일하다. 패킷 인터리버(1312)의 디멘젼은 포함된 메모리의 종류 및 크기에 따라 바뀔 수 있음에 유의해야 한다. 예컨대, 제1 디멘젼은 열들로 변경되고 제2 디멘젼은 행들로 변경될 수 있다. 또한 다른 디멘젼의 배열들이 사용될 수도 있다.

[0088] GF(256) SCBC(1314)는 앞서 설명한 바이트 코드 인코더들과 유사한 블록 코드 인코더이다. 특히, GF(256) SCBC(1314)는 갈루아 필드(256) 공간 상에서의 짧은 선형 블록 코드들을 이용하여 구현된다. 두개의 성분 블록 코드가 사용될 수 있다. 레이트 1/2 블록 코드 인코더는 다음 생성 행렬을 이용한다.

[0089] $$G = (1 \quad 2) \qquad (28)$$

[0090] 식 (28)의 행렬은 제2 열에 존재하는 식 (1)로부터 한 값을 갖는 b 성분을 포함한다. 레이트 2/3 블록 코드 인코더는 다음 생성 행렬을 이용한다.

[0091] $$G = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \end{pmatrix} \qquad (29)$$

[0092] 생성 행렬은 아이덴티티 행렬 및 b 성분들의 열을 이용하여 형성된다. 행렬 (29)에서 제3 열은 값 2를 갖는 식 (2) 및 (3)으로부터의 b 성분들을 포함한다. 각 성분 코드에 대한 생성 행렬에서의 계수들은 전체 오류 정정 시스템 및 변조 프로세스에 대한 블록 코드 인코딩의 관계에 기초하여 최적화되었다. 이 최적화는 특히 8-VSB

변조에서 트렐리스 코딩 및 비트 투 심볼(bit to symbol) 매핑을 고려하였는데 그 이유는 이런 성질들이 수신 및 복조 프로세스에서 제1 성질이기 때문이다.

[0093] GF(256) SCBC(1314)는 단일 또는 연쇄 블록 코드 인코더일 수 있다. 연쇄 블록 코드 인코더는 상기한 바와 같이 인터리빙 및 펑츄어링과 같은 다른 기능을 포함할 수 있다. GF(256) SCBC(1314)는 다중 인코딩된 레이트들을 인코딩할 수 있고 도시되지 않았지만 레이트 모드 제어기를 통해 레이트 모드들을 스위칭할 수도 있다. 바람직한 실시예에서, GF(256) SCBC(1314)는 상기 레이트 1/2 성분 코드, 도 6에 도시된 레이트 12/26 코드, 도 10에 도시된 레이트 12/52 코드, 또는 도 12에 도시된 레이트 17/26 코드 중 하나를 이용하여 데이터의 들어오는 스트림을 인코딩하도록 적응될 수 있다.

[0094] GF(256) SCBC(1314)는 인터리버(1312)로부터 출력된 열들을 따른 바이트를 인코딩한다. 즉, GF(256) SCBC(1314)는 패킷 인터리버(1312)에서의 처리를 통해 형성된 인터리버 행렬의 제2 디멘젼을 따라 인코딩한다.

[0095] 패킷 디인터리버(1316)는 GF(256) SCBC(1314)에 의해 생성된 코드워드들의 인코딩된 스트림을 수신하고 187 바이트 패킷들의 재구성 행들을 출력한다. 패킷 디인터리버(1316)는 인코딩된 코드워드들을 열별 순서로 입력하는데, 각 열은 GF(256) SCBC(1314)에서의 처리에 의해 추가된 리던던트나 비체계적 바이트를 포함하고, 이 바이트를 행별 배열로 출력한다. 이 프로세스는 근본적으로 패킷 인터리버(1312)에 대해 설명한 프로세스의 반대인데, 도 14 및 15의 순서를 역으로 한 것이다. 패킷 디인터리버(1312)는 동일 갯수의 코드워드들의 열들을 입력하는데, 각 코드워드는 비체계적 바이트의 인코딩된 집합을 이제 포함한다. 출력에서의 행들의 수는 인코딩된 코드워드 길이에 대응한다. 예컨대, 12/26 코드 레이트에서, 26 행들의 패킷들이 출력될 것이다. 패킷 디인터리버(1316)의 디멘젼은 포함된 메모리의 종류 및 크기에 기초하여 변할 수 있음에 유의한다. 또한, 제1 디멘젼은 행으로 바뀔 수 있고 제2 디멘젼은 열로 바뀔 수 있다. 또한 다른 디멘젼의 배열도 사용될 수 있다.

[0096] 패킷들은 두개의 별개의 그룹들로 배열될 수 있다. 제1 그룹의 패킷들은 체계적 패킷들이라 할 수 있고 트랜스포트 스트림 소스(1302)에 의해 제공된 원래 데이터 패킷들과 동일하다. 제2 그룹의 패킷들은 비체계적 패킷들이라 할 수 있고 GF(256) SCBC(1314)에서 블록 코딩 프로세스에 의해 형성된 패러티 패킷들이다. 블록 인코딩 프로세스의 결과, 열들의 수 (즉, 제2 디멘젼의 크기)가 증가하였음에 유의해야 한다.

[0097] MPEG 트랜스포트 스트림 헤더 수정기(1318)는 체계적 및 비체계적 패킷들의 그룹들을 포함하는 디인터리빙된 187 바이트 패킷들을 수신한다. 상술한 바와 같이, 각 패킷은 3 바이트 헤더를 포함한다. 3 바이트는 패킷에 대한 정보를 나르기 위해 사용되는 몇개의 다른 비트나 비트 그룹과 함께 PID도 포함한다. 레거시또는 A53 방송 신호를 수신하지만 ATSC M/H 인코딩된 패킷들을 올바로 디코딩할 수 없는 수신기 (예컨대, 레거시 수신기)의 가장 효율적인 동작을 유지하기 위해, ATSC M/H 패킷들의 일부의 헤더들에서의 특정 비트들이 수정될 수 있다. 비체계적 패킷 헤더들에서 이 비트들을 수정함으로써, 레거시 수신기들은 이 패킷들을 오류가 있는 것으로 간주하지 않으면서 이 패킷들을 무시하여야 한다. 예컨대, MPEG 트랜스포트 스트림 헤더 수정기(1318)는 TEI 비트, 페이로드 유닛 시작 표시 비트, 및 트랜스포트 우선순위 비트를 비트값 '0'으로 설정할 수 있다. 또한, 스크램블링 제어 및 적응(adaptation) 필드 비트들(각각 2비트)은 '00'으로 설정될 수 있다. 3비트 길이의 연속(countinuity) 카운터는 '000'으로 설정될 수 있다. 마지막으로, PID는 모든 레거시 수신기에 의해 무시될 알려진 값과 같은 고유하고 사용되지 않은 값으로 설정될 수 있다. MPEG 트랜스포트 스트림 헤더 수정기(1318)가 비체계적 패킷들의 그룹에 대한 각각의 헤더를 수정할 것이므로, GF(256) SCBC(134)가 비체계적 패킷 그룹에 대한 헤더들을 처리할 필요가 없을 수 있음에 유의해야 한다. 또한, MPEG 트랜스포트 스트림 헤더 수정기(1318)는 체계적 패킷들이 레거시 수신기에 의해 처리되지 않고 올바로 디코딩되지 않는다면 이 패킷들의 헤더들을 수정할 수 있다. 체계적 패킷들이 GF(256) SCBC 인코더(1314)에 의해 인코딩되지 않거나 MPEG 트랜스포트 스트림 헤더 수정기(1318)에 의해 처리되지 않으면, 결과적인 데이터 스트림은 모바일 장치 및 레거시 수신기의 양자로 동시 캐스팅되고 이에 의해 수신될 수 있다.

[0098] 아프리오리(a-priori) 트래킹 패킷 삽입기(1320)는 미리 정해진 트래킹 패킷들을 러기드 데이터 스트림 내에 위치시킬 수 있다. 미리 정해진 패킷들은 이동용, 보행자용 또는 휴대용 장치에서 사용되는 수신기와 같이 러기드 데이터 스트림을 수신할 수 있는 수신기에 완전히 또는 대부분 알려진 정보 패킷들을 나타낸다. 미리 정해진 패킷들은 신호 인코딩 및 전송의 레거시 또는 기존의 A53 인코딩 부분 동안에 생성된 트렐리스 상태를 디코딩할 때 지원하기 위해 수신기에서 사용된다. 미리 정해진 패킷들은 수신기의 이퀄라이저부에서 컨버전스(convergence)로 지원할 수도 있다. 미리 정해진 패킷들은 레거시 수신기에서의 수신을 개선하려는 것은 아니지만 결과적으로 잠재적인 개선을 가져올 수 있다. 또한, 종래의 트레이닝 정보와 달리, 미리 정해진 패킷들은 추가적인 레거시 인코딩이 수행되기 전에 추가되기 때문에 송신기 출력에서 직접 식별가능하지 않다. 특히, 미

리 정해진 패킷들은 트렐리스 인코딩의 처리에 의해 변경된다. 그 결과, 미리 정해진 패킷들은 트렐리스 디코딩 동안에 직접적인 트레이닝을 제공하기 보다는 트렐리스 디코딩 맵 또는 브랜치를 결정하는 데에 사용되는 아프리오리 브랜치 정보를 제공한다.

[0099] 미리 정해진 트래킹 패킷들은 알려진 트레이닝 시퀀스 프로세스를 이용하여 다양한 방법으로 생성될 수 있다. 바람직한 실시예에서, 미리 정해진 트래킹 패킷은 수신기에도 알려진 의사 난수 (PN) 생성기를 이용하여 생성되는 잔여 바이트를 갖는 유효 헤더를 포함한다. 아프리오리 트레이닝 데이터, 트렐리스 불명료 (trellis-obscured) 트레이닝 데이터, 또는 의사(pseudo) 트레이닝 패킷이라고도 지칭되는 미리 정해진 트래킹 패킷들은 ATSC M/H 전송 동안 몇몇 방식으로 분배될 수 있거나, ATSC M/H 신호 전송을 위한 프리앰블로서 기능하는 방식으로 패킷들 또는 패킷들의 그룹들을 배치하는 것을 포함하여 한 그룹 내에 클러스터링될 수 있다.

[0100] 레거시 ATSC 인코더(1350)는 상기한 바와 같이 레거시 A53 표준에 따라 체계적 패킷들 및 비체계적 패킷들을 동일하게 인코딩한다. 러기드 또는 로버스트 데이터 스트림을 생성할 ATSC M/H 블록(1310)의 추가 기능은 전송 설비의 기존 하드웨어 구조에 최소한의 변경을 가하여 추가될 수 있다. 또한, MPEG 트랜스포트 소스(1302)로부터의 들어오는 패킷들의 부분들은 ATSC M/H 블록(1310)에서 하나 또는 그 이상의 인코딩된 레이트로 인코딩하기 위해 추출될 수 있다. 인코딩된 패킷은 입력 패킷들의 나머지 미처리된 부분 내에 재삽입되거나 또는 이것에 부가될 수 있고 인코딩된 부분과 미처리된 부분의 양자 모두는 ATSC 레거시 인코더(1350)에서 인코딩될 수 있다. 이와 달리, 별개의 패킷들의 스트림은 ATSC M/H 블록(1310)에 제공될 수 있고 인코딩된 출력은 제2 패킷들의 스트림에 삽입 또는 부가되어 ATSC 레거시 인코더(1350)에 제공될 수 있다.

[0101] 도 16을 참조하면, 인코딩 프로세스(1600)의 일 실시예를 도시하는 흐름도가 도시되어 있다. 프로세스(1600)는 입력 데이터 스트림으로부터 러기드 데이터 스트림을 생성하기 위해 사용될 수 있는 연쇄 바이트 코드 인코딩 프로세스를 나타낸다. 프로세스(1600)는 주로 도 6에 도시된 연쇄 바이트 코드 인코더(600)를 참조하여 설명된다. 그러나, 이 프로세스는 도 6, 8, 9, 10 및 12와 앞서 설명된 인코더를 포함한 임의의 바이트 코드 인코더에 용이하게 적응될 수 있다. 프로세스(1600)는 이산 처리 블록들을 수반하는 하드웨어 또는 일부 또는 모든 필요한 블록을 포함하는 집적 회로를 사용하여, 마이크로프로세서 장치에서 동작하는 소프트웨어를 사용하여, 또는 하드웨어 및 소프트웨어의 조합을 이용하여 수행될 수 있음에 유의해야 한다. 또한, 프로세스(1600)는 바이트, 코드워드들 및 데이터 패킷들을 참조하여 설명될 것이다. 그러나, 다른 데이터 구성이나 배열도 가능하고 사용될 수 있음은 당업자에게 명백할 것이다.

[0102] 먼저, 단계 1610에서, 데이터 스트림이 수신된다. 데이터 스트림은 데이터의 바이트가 코드워드들로 그룹지어질 수 있고 하나 또는 그 이상의 코드워드의 전부나 일부를 포함하는 패킷들로 더 배열되는 식으로 배열될 수 있다. 예컨대, 데이터는 187 바이트의 데이터를 포함하는 패킷들로 배열될 수 있는데, 각 패킷은 식별 목적을 위해 사용되는 패킷 헤더를 포함한다. 다음에, 단계 1620에서, 데이터 패킷들은 바이트 코드 인코딩된다. 단계 1620에서의 인코딩은 앞서 논의한 성분 인코더들 중 하나를 이용하여 수행될 수 있다. 예컨대, 인코딩 단계 1620은 레이트 2/3 바이트 코드 인코딩을 이용할 수 있고 이것은 결과적으로 매 12 입력 바이트의 데이터에 대해 18 바이트의 데이터의 출력을 낳는다. 이와 달리, 인코딩 단계는 레이트 1/2와 같은 다른 바이트 코드 인코딩 레이트를 이용할 수 있다. 인코딩 단계 1620은 식(28) 및 식(29)에서 표시된 생성 행렬을 이용하여 입력 데이터 바이트를 보충(supplement)할 수 있다. 입력 데이터를 보충하는 것은 바이트 코드나 블록 코드 인코딩 프로세스와 같은 인코딩 프로세스를 통해 오류 정정 또는 리던던트 데이터 바이트를 생성하는 것을 포함한다. 출력 바이트는 6바이트의 리던던트 또는 비체계적 데이터와 함께 체계적 바이트로 알려진 12 바이트 입력 데이터의 사본을 포함한다.

[0103] 다음으로, 단계 1630에서, 단계 1620로부터의 인코딩된 데이터 바이트가 인터리빙된다. 몇개의 인터리빙된 배열이 사용될 수 있다. 예컨대, 도 7에 도시된 인터리빙된 배열이 사용될 수 있다. 도 7의 인터리빙된 배열은 바이트 코드 인코딩 단계 1620에서 생성된 코드들의 거리를 최대화하면서 비교적 작은 인터리버 크기를 제공한다. 즉, 인터리버 크기는 백색 잡음 (white noise)이 있는 경우에 바이트 오류 레이트를 줄이도록 최적화될 수 있다. 다음에, 단계 1640에서, 단계 1630으로부터의 인터리빙된 바이트가 두번째 바이트 코드 인코딩된다. 제2 바이트 코드 인코딩 단계 1640은 상술한 성분 인코더들 중 하나를 이용하여 수행될 수 있다. 예컨대, 단계 1620에서의 인코딩은 레이트 2/3 바이트 코드 인코딩을 이용할 수 있고 결과적으로 매 18 입력 데이터 바이트에 대해 27 바이트의 데이터의 출력을 낳는다. 이와 달리, 인코딩 단계는 레이트 1/2와 같은 다른 바이트 코드 인코딩 레이트를 이용할 수 있다. 인코딩 단계 1640은 상기한 바와 같이 식(28) 및 식(29)에 나타낸 생성 행렬을 이용하여 데이터의 입력 바이트를 보충할 수 있다. 출력 바이트는 9 바이트의 리던던트나 비체계적 데이터와 함께 체계적 바이트로 알려진 18 입력 바이트의 데이터의 사본들을 포함한다. 몇몇 체계적 바이트는 원래 입력

데이터의 바이트들 중 하나의 사본들일 수 있고 또는 제1 바이트 코드 인코딩 단계 1620에 의해 리던던트나 비체계적 바이트로 디벨롭되는 한 바이트일 수 있음은 물론이다.

[0104] 다음에, 단계 1650에서, 데이터의 바이트의 제2 인코딩된 스트림이 펑츄어링된다. 펑츄어링 단계 1650은 제2 인코딩된 스트림으로부터 데이터 바이트들 중 하나를 제거한다. 제거된 바이트는 제2 인코딩 단계 1640의 비체계적 바이트일 수 있고, 또한 제1 인코딩 단계 1620으로부터의 비체계적 바이트일 수 있다. 마지막으로, 단계 1660에서, 데이터 스트림이 레거시 또는 기존 A53 인코딩과 같은 추가 처리를 위해 제공된다. 단계 1660은 데이터 스트림을 제공하기 전에 원래 수신된 배열과 유사한 패킷들이 되도록 인코딩된 바이트를 다시 패킷타이징하는 단계를 포함할 수 있다. 상기 프로세스(1600)로 레이트 12/26 바이트 코드 인코딩된 데이터 스트림이 생성된다.

[0105] 단계 1650에서 펑츄어링은 프로세스(1600)로부터 제거될 수 있다. 제거를 위한 바이트의 선택은 단계 1630에서 인터리빙에 기초하여 수행된다. 예컨대, 제2 인코딩 단계 1640은 그 인코딩의 일환으로서 비체계적 바이트들 중 하나를 생성하지 않을 수 있어서 바로 펑츄어링된 스트림을 낳을 수 있다. 또한, 펑츄어링되지 않은 레이트 12/27 바이트 코드 인코딩된 데이터 스트림을 생성하기 위해 펑츄어링 단계 1650이 생략될 수도 있다.

[0106] 단계 1650에서의 펑츄어링은 제2 인코딩된 스트림으로부터 1 바이트 이상 제거할 수도 있다. 예컨대, 레이트 12/24 바이트 코드 인코딩된 데이터 스트림을 생성하기 위해 제거될 수 있는 3 바이트를 식별하는 것이 가능할 수 있다. 1 바이트 이상의 펑츄어링은 코딩 레이트에서의 개선을 얻어 내기는 하지만 인코딩의 유효성을 더 열화시킬 것이다. 펑츄어링 단계 1650에서의 추가 바이트의 제거는 단계 1630에서의 최적 인터리빙에 기초하여 달성된다. 이런 방식으로 펑츄어링 및 인터리빙은 출력 패킷들의 주어진 출력 블록 크기의 생성에 기초하여 최적의 코드 레이트를 허용하도록 상호작용한다.

[0107] 단계 1630 및 단계 1640은 두번의 인터리빙 단계 및 세번의 바이트 코드 인코딩 단계를 포함한 서로 다른 연쇄 바이트 코드 인코딩 프로세스를 형성하기 위해 반복될 수 있음에 유의한다. 반복 단계 1530 및 1540을 이용하는 프로세스는 레이트 12/52의 러기드 데이터 스트림을 생성하기 위해 도 9에 도시된 인코더(900)와 같은 인코더에 의해 사용될 수 있다. 프로세스(1600)는 앞서 설명한 것과 같은 다른 코드 레이트에 용이하게 적응될 수 있다.

[0108] 도 17을 참조하면, 인코딩 프로세스(1700)의 다른 실시예를 도시하는 흐름도가 도시되어 있다. 프로세스(1700)는 기존 또는 레거시 A53 신호 포맷에 따르는 체계적 및 비체계적 또는 리던던트 데이터 패킷을 포함하는 ATSC M/H 데이터 스트림을 인코딩하고 전송하는 단계를 나타낸다. 프로세스(1700)는 도 13에서의 인코더(1300)를 주로 참조하여 설명될 것이다. 상기한 바와 같이, 프로세스(1700)는 일부 또는 모든 필요한 블록을 포함하는 집적 회로나 이산 처리 블록을 수반하는 하드웨어, 마이크로프로세서 장치에서 동작하는 소프트웨어, 또는 하드웨어와 소프트웨어의 조합을 이용하여 수행될 수 있다. 프로세스(1700)는 요구되는 구현에 기초하여 특정 단계를 제거 또는 재배열함으로써 적응될 수 있음에 유의한다.

[0109] 먼저, 단계 1710에서, 패킷들의 전송 스트림이 수신된다. 각 패킷은 187 바이트를 포함하고 헤더를 포함한다. 헤더는 PID 뿐만 아니라 패킷에 관한 다른 정보를 포함한다. 다음에, 단계 1720에서, ATSC M/H 패킷들에 대해 사용되는 것으로 식별되는 패킷들이 분리되거나 추출된다. 나머지 패킷들은 미처리로 식별된다. ATSC M/H 패킷들은 조합된 단일 트랜스포트 스트림으로부터 추출되는 대신에 패킷들의 별개의 입력 트랜스포트 스트림으로서 제공될 수 있다. 또한, 트랜스포트 스트림에서의 모든 패킷들은 ATSC M/H 패킷들로 식별될 수 있다. 이 조건들 중 어느 것이라도 추출 단계 1720의 필요성을 제거할 수 있다. 또한, ATSC M/H이나 미처리로 식별된 패킷들은 그룹지어질 수 있고 ATSC M/H 식별된 패킷들은 별개의 인코딩 코드 레이트들에 의해 더욱 식별되어 그룹지어질 수 있다.

[0110] 다음에, 단계 1730에서, ATSC M/H 식별된 패킷들의 집합들 또는 그룹들은 판독되고 또는 행들로 입력되고 열들로 출력되고 또는 패킷 인터리빙된다. 출력 데이터의 열들은 각 코드워드의 크기가 패킷들의 그룹의 크기와 동일하게 되면서 코드워드들에 동등하다. 도 14 및 도 15는 단계 1730에서 행들로 판독하고 열들을 출력하는 패킷 인터리빙을 나타내는 행렬을 도시한다. 단계 1730에서 사용된 인터리버의 디멘젼은 예컨대 열들을 입력하고 행들을 출력하는 것으로 변경되거나 또는 인터리버 구현에 기초한 임의의 다른 디멘젼 양태들을 이용하도록 변경될 수 있음에 유의한다. 다음에, 단계 1740에서, 단계 1730으로부터의 각 코드워드가 블록 코드 인코딩된다. 단계 1730에서의 블록 코드 인코딩은 프로세스(1600)에서의 바이트 코드 인코딩과 유사하고 간단한 인코딩 프로세스나 연쇄 인코딩 프로세스를 이용할 수 있다. 예컨대, 블록 코드 인코딩 단계 1730은 레이트 1/2 구성 코드, 레이트 12/26 코드, 레이트 12/52 코드 또는 레이트 17/26 코드를 이용하여 코드워드들을 인코딩할 수 있

다.

[0111] 다음에, 단계 1750에서, 인코딩된 코드워드들의 집합은 코드워드들을 열들로서 입력하고 데이터 패킷들을 행들로서 출력함으로써 패킷 디인터리빙된다. 입력 코드워드들은 단계 1730에서 블록 코드 인코딩에 의해 생성된 바이트의 수를 포함한다. 출력 패킷들은 187 바이트를 포함하는 패킷들로 재구성된다. 블록 코드 인코딩 단계 1730에서 생성된 비체계적 바이트는 인코딩된 데이터 스트림에서 추가 패킷들의 행들을 구성한다. 단계 1760에서 사용되는 디인터리버의 디멘젼은 예컨대 열들을 입력하고 행들을 출력하는 것으로 변경되거나 인터리버 구현에 기초한 임의의 다른 디멘젼 양태들을 사용하도록 변경될 수 있음에 유의해야 한다.

[0112] 다음에, 단계 1760에서, 인코딩된 디인터리빙 패킷들에서의 헤더 바이트가 변경된다. 변경 단계 1760은 헤더 정보가 레거시 수신기에 의해 인식되지 못하게 함으로써 ATSC M/H 데이터 패킷들을 디코딩할 수 없는 수신기에서의 성능 문제를 방지하는 방법을 제공한다. 단계 1760에서의 변경은 TEI 비트, 페이로드 유닛 시작 표시 비트 및 전송 패러티 비트를 비트값 '0'으로 설정하는 단계를 포함할 수 있다. 단계 1760에서의 변경은 또한 스크램블링 제어 및 적응 필드 비트들 (각각 2 비트)을 '00'으로 설정하는 단계를 포함할 수 있다. 변경 단계 1760은 또한 3 비트인 연속 카운터를 '000'으로 설정하는 단계를 포함할 수 있다. 마지막으로, 단계 1760에서의 변경은 모든 레거시 수신기에 의해 무시될 알려진 값과 같은 고유하고 미사용된 값으로 PID를 설정하는 단계를 포함할 수 있다. 헤더 바이트가 무시될 수 있고 인코딩 단계 1640에서 처리되지 않을 수 있음을 유의해야 한다.

[0113] 단계 1770에서, 미리 정해진 패킷들이나 아프리오리 트래킹 패킷들은 변경된 헤더 정보를 포함하는 인코딩된 패킷들의 스트림내로 삽입된다. 아프리오리 트래킹 패킷들의 삽입은 ATSC M/H 또는 모바일 영상 인코딩된 신호를 수신할 수 있는 수신기의 성능을 개선한다. 삽입 단계 1770는 기존의 리던던트 또는 비체계적 패킷을 대체하거나, 데이터 패킷들의 스트림에서의 널(null) 패킷으로서 단계 1710에서 원래 제공된 패킷을 대체할 수 있음에 유의해야 한다.

[0114] 단계 1780에서, 단계 1770으로부터의 ATSC M/H 인코딩된 패킷들은 데이터의 트랜스포트 스트림의 미처리 부분과 조합된다. ATSC M/H 인코딩된 패킷들은 데이터 패킷들의 트랜스포트 스트림의 이전에 식별된 미처리된 부분에 삽입되거나 부가될 수 있다. 이와 달리, 단계 1770으로부터의 ATSC M/H 인코딩된 패킷들은 레거시 방송 수신만을 위해 식별되는 제2 전송 스트림과 조합되거나, 이에 삽입되거나, 또는 부가될 수 있다. 단계 1710에서의 모든 패킷이 ATSC M/H 데이터 패킷들로서 식별되어 처리되었다면 단계 1780은 제거될 수도 있음에 유의해야 한다. 다음에, 단계 1790에서, ATSC M/H 인코딩되거나 되지 않은 모든 패킷을 포함한 완전한 데이터 스트림이 A53 표준에 따르는 레거시 또는 기존의 인코딩을 이용하여 처리된다. 단계 1790에서의 레거시 인코딩은 리드-솔로몬 인코딩, 랜더마이징, 인터리빙, 트렐링(trelling) 인코딩 및 동기화 삽입을 포함한다. 레거시 인코딩 단계 1790은 레거시 인코더(1350)에서 도시된 것과 같은 블록에 의해 수행될 수 있다.

[0115] 마지막으로, 단계 1795에서, ATSC M/H 데이터로서 인코딩된 모든 또는 일부 스트림을 포함한 완전히 인코딩된 데이터 스트림이 전송된다. 전송 단계 1795는 특정하게 식별된 주파수 범위를 이용하여 전송하는 단계를 포함할 수 있고 동축 케이블과 같은 유선 기술을 이용하거나 전자기적으로 공중파를 통해 전송하는 단계를 포함할 수 있다. ATSC M/H 데이터가 연속적으로 전송될 수 있음에 유의해야 한다. 이 모드에서, ATSC M/H 체계적 패킷은 또한 레거시 수신기에서 데이터 패킷들로서 기능한다. 비체계적 패킷들은 무시될 것이다. 그러나, 별개의 ATSC M/H 및 레거시 데이터는 ATSC M/H 데이터가 주기적으로 전송되거나 짧은 비연속 시구간들 동안 연속적으로 전송되는 방식으로 전송될 수 있다.

[0116] 도 18을 참조하면, 테이블(1800)은 심볼의 전송 포맷의 비트들로의 매핑을 도시한다. 테이블(1800)은 전송 데이터의 두 개의 비트 Z1 및 Z2를 나타내는 심볼들 0-4 세트의 매핑을 도시한다. 이 매핑은 각 심볼에 대응하는 신호 진폭을 4-PAM 신호로서 변조함으로써 전송될 수 있다. 심볼의 최상위 비트 (Z2)에 대한 두 개의 값 또는 상태 사이의 평균으로서의 전압 또는 진폭 차이는, 심볼의 최하위 비트 (Z1)에 대한 두 개의 값 또는 상태 사이의 전압 또는 진폭 차이보다 훨씬 더 크다. 그 결과, 특정 심볼에 대응하는 신호가 잡음이 심한 전송 채널을 통해 전송될 때, 수신기에서 Z2 비트를 올바로 추정할 확률은 Z1 비트를 올바로 추정할 확률보다 크다. 데이터 심볼 및 이 데이터 심볼의 주기적 순환(a cyclic rotation)인 제2 심볼을 전송하면 전송된 심볼의 Z1 및 Z2 비트를 올바로 복구할 확률을 크게 증가시킬 수 있는 코드를 산출할 수 있다. 수학식 (1) 내지 (27)에서 사용된 갈루아 필드의 성분 b_i 값들 및 수학식 (28) 및 (29)에서 사용된 생성 행렬들은 패킷을 포함하는 바이트의 비트들을 주기적으로 순환 (및/또는 랩(wrap))하도록 선택될 수 있고 따라서 이러한 수학식들을 이용하는 바이트 코드 인코더들에 의해 생성된 데이터 스트림의 강건한 특성을 더 개선할 수 있다. 주기적 순환의 선택은, 바이트

의 마지막 비트가 갈루아 필드의 특정 특성에 기초하여 제1 바이트로 랩 어라운드(wrap around)될 때 간단한 주기적 순환으로부터 수정되는 결과를 가져올 수 있다. 도 1에 도시된 것과 같은 트렐리스 인코더에 의한 추가 처리는 원래의 비트 투 심볼 매핑에 대해 크게 영향을 미치지 않으면서 전송된 신호에 리던던트 정보를 더 추가할 것임에 유의해야 한다. 트렐리스 인코딩 및 추가 신호 필터링은 여기 설명된 4-PAM 신호를 A53 표준에서 설명된 8-VSB 신호로 변환하는 결과를 가져올 것이다.

[0117] 도 19를 참조하면, 컨볼루션 인터리버에서 처리되는 바이트의 인터리버 맵(1900)을 나타내는 도면이 도시된다. 인터리버 맵(1900)은 도 13의 ATSC 레거시 인코더(1350)에서 사용되는 인터리버(1356)와 같은 컨볼루션 인터리버의 처리 동안에 데이터의 들어오는 바이트의 조직을 도시한다. 인터리버(1356)가 일련의 지연 라인들을 이용하여 구현될 수 있지만, 인터리버 맵(1900)은 인터리버를 위한 메모리 맵이라고 고려될 수 있다. 인터리버 맵(1900)은 배치되거나 기입되는 입력 바이트의 위치와 출력 바이트의 판독 방법을 표시한다. 인터리버 맵(1900)의 디멘젼은 상단의 0 부터 206 까지의 바이트 및 측면의 위에서 아래로 0 부터 103 까지의 행으로 표시된다. 라인 1910은 바이트가 판독 출력되는 순서를 나타낸다. 예컨대, 라인 1910이 행 20을 나타내면, 행 20에 있는 모든 바이트가 바이트 0에서 시작하여 바이트 206에서 끝날 때까지 판독 출력될 것이다. 마지막 바이트 206이 행 20으로부터 판독 출력될 때, 판독이 한 행 더 앞으로 나가 행 21로 진행하고, 이는 인터리버의 마지막 행이 판독될 때까지 계속된다. 마지막 행이 판독 출력될 때, 첫 행 판독으로 (새로운 패킷 데이터로) 다시 시작한다.

[0118] 라인 1920은 207 바이트 코드 인코딩된 제1 패킷의 처음 52 바이트와 리드-솔로몬 바이트의 위치를 나타내고, 이는 인터리버로의 이 바이트들의 판독에 기초한다. 라인 1920은 패킷에서 바이트 0의 위치에서 시작하고 바이트 51의 위치에서 중앙 라인 1990에서 종료한다. 라인 1922, 1924, 1926a 및 1926b는 제1 패킷에서 나머지 바이트의 위치를 보여준다. 라인 1922는 라인의 상단의 바이트 52의 위치에서 시작하고, 이런 식으로 계속되며, 각 라인 1922, 1924 및 1926a에 대한 바이트 위치들에서 처리한다. 바이트의 나머지 부분은 라인 1926b를 따라 위치하고 라인 1990의 한 행 아래의 행의 한 위치에서 바이트 206에서 종료한다. 연속하는 패킷들에서 바이트의 위치는 제1 패킷에 대한 위치들의 오른쪽으로 계속되고 라인 1990 위의 진행 및 위치를 미러링(mirroring)하는 라인 1990 아래의 맵의 부분으로 진행한다. 예컨대, 라인 1930은 인터리버에서 52번째 패킷 (즉 제1 패킷 이후의 패킷 입력 52 패킷들)에 대한 바이트의 일부의 위치를 나타낸다. 라인 1950은 패킷 그룹의 전송을 위한 경계선을 나타낸다. 각 연속하는 패킷에서, 그 패킷으로부터 다음 연속하는 바이트가 경계선에 닿는다(fall on). 그 결과, 라인 1950은 패킷 0 바이트 0 위치, 패킷 1 바이트 1 위치, 이렇게 계속하여 패킷 52 바이트 52 위치까지 나타낸다. 라인 1960은 라인 1950과 라인 1920 사이의 한 행에 있는 바이트의 위치를 나타낸다. 특히, 라인 1960은 도 14에 도시된 바와 같이 행들로 배향된 패킷 세트의 열 26으로부터의 바이트들인 바이트 그룹의 위치를 나타낸다. 다음 행의 바이트는 이 패킷 집합의 일부의 바이트 27을 포함한다. 바이트 코드 인코딩된 바이트가 인터리버로부터 짧은 연속 시구간 동안 그룹들로 출력되어 전송될 것이므로 라인 1960은 상기 바이트 코드 인코딩 프로세스를 이용하여 데이터를 인코딩하는 이점을 도시한다.

[0119] 패킷 세트의 특정 배열은 로버스트 데이터 스트림의 보다 최적의 전송 배열을 제공할 수 있음에 유의한다. 로버스트 데이터 스트림의 배열은 로버스트 데이터 스트림이 연속적으로 전송되지 않은 경우에 (즉, 데이터 스트림의 일부가 레거시 데이터인 경우) 중요할 수 있다. 예컨대, 도 19에 도시된 바와 같이, 52 패킷의 세트는 결과적으로 수신 시스템에서 디인터리빙 프로세스를 이용하여 용이하게 예측되고 식별될 수 있는 방식으로 로버스트 데이터를 전송하는 것을 포함하는 전송 특성으로 된다.

[0120] 도 20을 참조하면, 수신기 시스템에서 사용되는 디코더(2000)의 일 실시예의 블록도가 도시되어 있다. 디코더(2000)는 수신기에 의해 수신되는 데이터를 디코딩할 때 돕기 위하여 앞서 설명한 바와 같이 데이터 스트림에서 비체계적 패킷과 같은 리던던트 패킷을 사용하도록 적응되는 회로 소자를 포함한다. 디코더(2000)는 일반적으로 레거시 또는 기존 A53 표준을 이용하여 인코딩된 데이터를 디코딩할 수 있다.

[0121] 디코더(2000)에서, 초기 튜닝, 복조 및 다른 회로 (도시되지 않음)에 의한 처리 이후, 트렐리스 디코더(2002)는 들어오는 신호를 수신한다. 트렐리스 디코더(2002)는 컨볼루션 디인터리버(2004)에 연결된다. 컨볼루션 디인터리버(2004)의 출력은 바이트 코드 디코더(2006)에 연결된다. 바이트 코드 디코더(2006)는 리드-솔로몬 디코더(2008)에 연결된 출력을 갖는다. 리드-솔로몬 디코더(2008)의 출력은 디랜더마이저(2010)에 연결된다. 디랜더마이저(2010)의 출력은 데이터 디코더(2012)에 연결된다. 데이터 디코더(2012)는 비디오 디스플레이 또는 오디오 재생과 같은 수신기 시스템의 나머지 부분에서 사용을 위한 출력 신호를 제공한다. 트렐리스 디코더(2002), 디인터리버(2004), 리드-솔로몬 디코더(2008), 디랜더마이저(2010) 및 데이터 디코더(2012)는 도 2에

도시된 블록과 기능이 유사하므로 더 이상 상술하지 않는다.

[0122] 데이터 패킷들에서 데이터의 바이트 형태로 수신된 데이터는 트렐리스 디코더(2002)에 의해 디코딩되고 디인터리버(2004)에 의해 디인터리빙된다. 데이터 패킷들은 207 바이트의 데이터를 포함할 수 있고 그룹들이나 24, 26 또는 52 패킷으로 더 그룹지어질 수 있다. 트렐리스 디코더(2002) 및 디인터리버(2004)는 레거시 포맷의 들어오는 데이터는 물론 바이트 코드 인코딩된 데이터를 처리할 수 있다. 수신기에 의해 알려져 있는 미리 정해진 패킷 전송 시퀀스에 기초하여, 바이트 코드 디코더(2006)는 패킷이 바이트 코드 인코딩된 또는 로버스트 데이터 스트림에 포함된 패킷인지 결정한다. 수신된 패킷이 바이트 코드 인코딩된 데이터 스트림으로부터가 아니라면, 수신된 패킷은 바이트 코드 디코더(2006)에서의 임의의 추가 처리 없이 리드-솔로몬 디코더(2008)로 제공된다. 바이트 코드 디코더(2006)는 인코딩 중에 데이터 스트림에 의해 곱해지거나 이것에 가산되는 알려진 상수 시퀀스를 제거하는 디랜더마이저를 포함할 수도 있다. 러기드 데이터 스트림은 원 데이터와 동일한 바이트와 체계적 패킷들 및 리던던트 데이터를 포함하는 바이트와 비체계적 패킷들을 모두 포함함에 유의한다.

[0123] 바이트 코드 디코더(2006)가 수신된 패킷이 로버스트 또는 러기드 데이터 스트림에 속하는 바이트 코드 인코딩된 패킷이라고 결정하면, 패킷은 동일한 데이터 스트림을 포함하는 다른 패킷들과 함께 디코딩될 수 있다. 일실시예에서, 동일 데이터 스트림의 바이트 코드 인코딩된 패킷들은 패킷내 각 바이트를 상기 식(2)에 도시된 바와 같이 바이트 코딩된 패킷을 디벨롭하도록 사용된 b 성분의 값의 역(inverse)으로 곱함으로써 디코딩된다. 비체계적 패킷의 바이트의 디코딩 값은 체계적 패킷의 바이트의 값에 비교되고, 동일하지 않은 두개의 패킷에서 임의의 바이트의 값들은 체계적 패킷에서 제거(즉, 0으로 설정)되거나 비체계적 패킷에서의 정보에 의해 대체될 수 있다. 오류 바이트가 제거된 체계적 패킷은 그후 리드-솔로몬 디코더(2008)에서 수행되는 리드-솔로몬 디코딩을 이용하여 디코딩될 수 있다. 바이트 코드 디코더의 다른 실시예에 대한 추가 설명이 이하에서 논의될 것이다.

[0124] 바이트 코드 디코더(2006)는 도 13에 도시된 바와 같이 인코딩된 신호를 디코딩하기 위한 블록 코더로서 동작하도록 적응될 수 있다. 예컨대, 바이트 코드 디코더(2006)는 패킷 인터리버(1312)와 유사한 패킷 인터리버 및 패킷 디인터리버(1316)와 유사한 패킷 디인터리버를 포함할 수 있다. 또한, 바이트 코드 인코더 기능은 GF(256) SCBC (Serial Concatenated Block Coded) 신호를 디코딩하도록 적응될 수 있다. 바이트 코드 디코더(2006)는 모바일 또는 ATSC M/H 수신을 위해 인코딩되는 데이터의 식별 및/또는 아프리오리 트레이닝 패킷의 식별을 위해 사용되는 식별 블록을 더 포함할 수 있다. 또한, 식별 블록은 예컨대 들어오는 패킷들의 헤더가 모바일 수신을 위해 사용되는 PID를 포함하는지 여부를 결정하기 위해 패킷 식별 블록을 포함할 수 있다.

[0125] 도 5에 도시된 인코더(500)와 같은 바람직한 인코더에서, 바이트 코드 인코딩은 데이터 패킷들의 리드-솔로몬 인코딩에 우선함에 유의해야 한다. 그러나, 여기 도시된 디코더(2000)에서, 들어오는 데이터는 리드-솔로몬 디코딩 전에 바이트 코드 디코딩된다. 바이트 코드 동작과 리드-솔로몬 코드 동작이 A53 표준에서 사용되는 갈루아 필드(256)에 대해서 선형이고 선형 연산자가 갈루아 필드에서 가환이기 때문에 재순서화(re-ordering)가 가능하다. 재순서화의 중요성은 바이트 코드 인코딩이 수신 신호에서의 오류를 복구하는 데에 더 높은 신뢰도를 제공하는 것에 있다. 그 결과, 리드-솔로몬 디코딩 이전에 바이트 코드 디코딩을 수행하는 것은 비트 오류율 및 신호 대 잡음비의 면에서 볼 때 수신기 성능이 개선되게 한다.

[0126] 또한, 도 5의 인코더(500)와 도 13의 인코더(1300)의 실시예에 대한 설명으로, 바이트 코드 디코딩 요소들과 프로세스는 레거시 수신기에 최소한의 변경으로 레거시 수신기에서 필요하고 사용되는 요소들, 컴포넌트들 및 회로들에 부가될 수 있다. 그러나, 디코딩 프로세스는 바이트 코드 디코딩 프로세스의 특징을 레거시 수신기의 다른 블록 내에 포함시킴으로써 향상될 수 있다.

[0127] 도 21을 참조하면, 수신기에서 사용되는 디코더(2100)의 다른 실시예의 블록도가 도시되어 있다. 디코더(2100)는 공중을 통한 전자기파와 같은 전송 매체를 통해 신호의 전송에 의해 악영향을 받은 신호를 수신하여 디코딩하기 위한 추가적인 회로 소자 및 처리를 포함한다. 디코더(2100)는 레거시 데이터 스트림 뿐만 아니라 러기드 데이터 스트림도 디코딩할 수 있다.

[0128] 디코더(2100)에서, 들어오는 신호가 초기 처리 이후 이퀄라이저(2106)로 제공된다. 이퀄라이저(2106)는 두개의 출력을 제공하는 트렐리스 디코더(2110)에 연결된다. 트렐리스 디코더(2110)로부터의 제1 출력은 피드백을 제공하고 이퀄라이저(2106)에 피드백 입력으로서 다시 연결된다. 트렐리스 디코더(2110)로부터의 제2 출력은 컨볼루션 디인터리버(2114)에 연결된다. 컨볼루션 디인터리버(2114)는 두개의 출력을 역시 제공하는 바이트 코드 디코더(2116)에 연결된다. 바이트 코드 디코더(2116)로부터의 제1 출력은 컨볼루션 인터리버(2118)를 통해 트렐리스 디코더(2110)에 피드백 입력으로 다시 연결된다. 바이트 코드 디코더(2116)로부터의 제2 출력은 리드-

솔로몬 디코더(2120)에 연결된다. 리드-솔로몬 디코더(2120)의 출력은 디랜더마이저(2124)에 연결된다. 디랜더마이저(2124)의 출력은 데이터 디코더(2126)에 연결된다. 리드-솔로몬 디코더(2120), 디랜더마이저(2124), 및 데이터 디코더(2126)는 도 2에서 설명한 리드-솔로몬, 디랜더마이저 및 데이터 디코더 블록과 유사한 방식으로 연결되어 기능적으로 동작하고 이에 대해서는 더 설명하지 않는다.

[0129] 수신기 (도시되지 않음)의 프론트엔드 처리(예컨대, 안테나, 튜너, 복조기, A/D 변환기)로부터의 입력 신호는 이퀄라이저(2106)로 제공된다. 이퀄라이저(2106)는 수신 신호를 복구하도록 전송 채널 효과를 완전히 또는 부분적으로 제거하기 위해 수신 신호를 처리한다. 다양한 제거 또는 이퀄라이징 방법이 당업자에게 잘 알려져 있으므로 여기서는 논의하지 않는다. 이퀄라이저(2106)는 피드 포워드 이퀄라이저 (FFE) 섹션 및 결정 피드백 이퀄라이저 (DFE) 섹션을 포함하는 다수 섹션의 처리 회로 소자를 포함할 수 있다.

[0130] 이퀄라이징된 신호는 트렐리스 디코더(2110)로 제공된다. 트렐리스 디코더(2110)는 이퀄라이저(2106)의 DFE 섹션에 제공되는 결정값 세트를 일 출력으로서 생성한다. 트렐리스 디코더(2110)는 이퀄라이저(2106)의 DFE 섹션에 역시 제공되는 중간 결정값들을 생성할 수 있다. DFE 섹션은 이퀄라이저(2106)에서 필터 탭의 값을 조정하기 위해 트렐리스 디코더(2110)로부터의 중간 결정값들과 함께 결정값들을 사용한다. 조정된 필터 탭 값은 수신된 신호에 존재하는 간섭 및 신호 반사를 제거한다. 반복적인 프로세스는 트렐리스 디코더(2110)로부터의 피드백의 도움으로 이퀄라이저(2106)가 시간에 따라 잠재적으로 변화하는 신호 전송 환경 조건에 동적으로 맞출 수 있게 한다. 반복적인 프로세스는 디지털 텔레비전 방송 신호에 대한 19 Mb/s와 같은 신호의 들어오는 데이터 레이트와 유사한 레이트로 발생할 수 있음에 유의한다. 반복적인 프로세스는 들어오는 데이터 레이트보다 더 높은 레이트로 발생할 수도 있다.

[0131] 트렐리스 디코더(2110)는 트렐리스 디코딩된 데이터 스트림을 컨볼루션 디인터리버(2114)로 제공한다. 컨볼루션 디인터리버(2114)는 도 20에서 설명된 디인터리버와 유사하게 동작하고 데이터 패킷 내에서 조직된 디인터리빙된 바이트를 생성한다. 데이터 패킷은 바이트 코드 디코더(2116)에 제공된다. 상기한 바와 같이, 러기드 데이터 스트림의 일부가 아닌 패킷은 단순히 바이트 코드 디코더(2116)를 통해 리드-솔로몬 디코더(2120)로 전달된다. 바이트 코드 디코더(2116)가 패킷 그룹을 러기드 데이터 스트림의 일부로서 식별하면, 바이트 코드 디코더(2116)는 비체계적 패킷 내의 리던던트 정보를 사용하여 앞서 설명된 바와 같이 초기에 패킷들내의 바이트를 디코딩한다.

[0132] 바이트 코드 디코더(2116) 및 트렐리스 디코더(2110)는 러기드 데이터 스트림을 디코딩하기 위해 터보 디코더라고 하는 반복적인 방식으로 동작한다. 구체적으로, 트렐리스 디코더(2110)는 컨볼루션 디인터리버(2114)에 의한 디인터리빙 후에 러기드 데이터 스트림에 포함된 패킷들의 각 바이트에 대해 제1 연판정(soft decision) 벡터를 바이트 코드 디코더(2116)에 제공한다. 일반적으로, 트렐리스 디코더(2110)는 확률값들의 벡터로서 연판정을 생성한다. 일부 실시예에서, 벡터에서의 각 확률값은 이 벡터와 연관된 바이트가 가질 수 있는 값과 연관된다. 다른 실시예에서, 확률값들의 벡터는 2/3 레이트 트렐리스 디코더가 2비트 심볼을 추정하기 때문에 체계적 패킷에 포함된 매 1/2 니블 (즉, 2 비트)에 대해 생성된다. 일부 실시예에서, 트렐리스 디코더(2110)는 바이트가 가질 수 있는 값들의 확률들의 벡터인 하나의 연판정을 생성하기 위해 한 바이트의 네 개의 1/2 니블과 연관된 4개의 연판정을 조합한다. 이러한 실시예에서, 바이트에 대응하는 연판정들이 바이트 코드 디코더(2116)에 제공된다. 다른 실시예에서, 바이트 코드 디코더는 체계적 패킷의 1 바이트에 대한 연판정을 4개의 연판정 벡터로 분리하는데, 여기서 각 연판정은 바이트의 1/2 니블과 연관된다.

[0133] 바이트 코드 디코더(2116)는 러기드 데이터 스트림의 패킷들을 포함하는 바이트와 연관된 연판정 벡터를 이용하여 이 패킷들을 포함하는 바이트의 제1 추정을 산출한다. 바이트 코드 디코더(616)는 체계적 및 비체계적 패킷을 이용하여 러기드 스트림을 포함하는 패킷들의 각 바이트에 대한 제2 연판정 벡터를 생성하고 컨볼루션 인터리버(2118)에 의한 리인터리빙 후에 제2 연판정 벡터를 트렐리스 디코더(2110)에 제공한다. 트렐리스 디코더(2110)는 그후 제2 연판정 벡터를 이용하여 제1 연판정 벡터의 추가적인 반복을 생성하고, 이는 바이트 코드 디코더(2116)로 제공된다. 트렐리스 디코더(2110) 및 바이트 코드 디코더(2116)는 트렐리스 디코더와 바이트 코드 디코더에 의해 생성된 연판정 벡터가 수렴하거나 소정 횟수의 반복이 수행될 때까지 이런 방식으로 반복한다. 그후, 바이트 코드 디코더(2116)는 체계적 패킷의 각 바이트에 대한 연판정 벡터에서의 확률값들을 이용하여 체계적 패킷들의 각 바이트에 대해 경판정(hard decision)을 생성한다. 경판정값들(즉, 디코딩된 바이트)은 바이트 코드 인코더(2116)로부터 리드-솔로몬 디코더(2120)로 출력된다. 트렐리스 디코더(2110)는 MAP (Maximum a Posteriori) 디코더를 이용하여 구현될 수 있고 바이트나 1/2 니블 (심볼) 연판정들에서 동작할 수 있다.

[0134] 터보 디코딩은 일반적으로 들어오는 데이터 레이트들 보다 더 높은, 블록들 사이의 결정 데이터의 전달과 연관된 반복 레이트들을 이용함에 유의한다. 가능한 반복의 수는 데이터 레이트 및 반복 레이트의 비에 제한된다. 그 결과 및 실제적인 범위에서, 터보 디코더에서의 더 높은 반복 레이트는 일반적으로 오류 정정 결과를 개선한다. 일 실시예에서, 들어오는 데이터 레이트의 8배인 반복 레이트가 사용될 수 있다.

[0135] 도 21에서 설명된 것과 같은 SISO (soft input soft output) 바이트 코드 디코더는 벡터 디코딩 기능을 포함할 수 있다. 벡터 디코딩은 체계적 및 비체계적 바이트를 포함하는 데이터의 바이트를 그룹짓는 것을 포함한다. 예컨대, 레이트 1/2 바이트 코드 인코딩된 스트림의 경우, 1 체계적 및 1 비체계적 바이트가 그룹지어질 것이다. 두 개의 바이트는 64,000개 이상의 가능한 값을 가진다. 벡터 디코더는 두 개의 바이트의 가능한 값들 각각에 대한 확률을 결정 또는 추정하고 확률 맵을 생성한다. 연판정은 일부 또는 모든 가능성들의 확률의 가중 및 가능한 코드워드까지의 유클리드 거리에 기초하여 이루어진다. 경판정은 유클리드 거리의 에러가 한계값 아래로 떨어질 때 이루어질 수 있다.

[0136] 도 20 및 21에서 설명된 바이트 코드 디코더는 단순한 바이트 코드 인코더나 연쇄 바이트 코드 인코더에 의한 인코딩을 포함하여 상기 바이트 코드 인코더에 의해 인코딩된 러기드 데이터 스트림을 디코딩할 수 있다. 도 20 및 21의 바이트 코드 디코더는 단일 인코딩 단계만을 포함하는 단순 또는 성분 바이트 코드 인코더에 의해 인코딩되는 러기드 데이터 스트림을 디코딩하는 것을 설명한다. 연쇄 바이트 코드 디코딩은 디인터리빙, 디펑츄어링 및 재삽입과 같은 중간 처리 이외에 하나 이상의 디코딩 단계에서 들어오는 코드워드나 바이트를 디코딩하는 것을 포함한다.

[0137] 도 22를 참조하면, 연쇄 바이트 코드 디코더(2200)의 일 실시예의 블록도가 도시되어 있다. 연쇄 바이트 코드 디코더(2200)는 도 21에 도시된 것과 같은 터보 디코더 구성에서 동작하도록 구성된다. 연쇄 바이트 코드 디코더(2200)는 또한 러기드 데이터 스트림에서 연쇄 바이트 코드 인코딩 패킷을 디코딩하기 위해 반복 프로세스를 이용하는 터보 디코더로서 내부적으로 동작한다. 연쇄 바이트 코드 디코더(2200)는 레이트 12/26 바이트 코드 인코딩된 신호 스트림을 디코딩하도록 적응되어 원래 인코딩된 26 바이트로부터 12 바이트의 데이터를 생성한다.

[0138] 26 바이트의 연판정값들을 나타내는 데이터 스트림이 바이트 삽입 블록(2202)에 제공된다. 바이트 삽입 블록(2202)의 출력은 제1의 2/3 레이트 바이트 코드 디코더(2204)에 연결된다. 제1의 2/3 레이트 바이트 코드 디코더(2204)는 두개의 출력을 제공한다. 제1 출력은 펑츄어 블록(2206)에 연결되고, 펑츄어 블록의 출력은 도 21에 도시된 바와 같이 인터리버를 통해 트렐리스 디코더에 피드백 입력으로 연결된다. 제1의 2/3 레이트 바이트 코드 디코더(2204)의 제2 출력은 디인터리버(2208)에 연결된다. 심볼 디인터리버(2208)의 출력은 역시 두개의 출력을 갖는 제2의 2/3 레이트 디코더(2210)에 연결된다. 제1 출력은 인터리버(2212)를 통해 제1의 2/3 레이트 바이트 코드 디코더(2204)에 피드백 입력으로 연결된다. 제2 출력은 리드-솔로몬 디코더와 같은 다른 처리 블록에 연결된다.

[0139] 바이트 삽입 블록(2202)으로 입력되는 26 바이트는 데이터의 체계적 바이트나 체계적 패킷에 관한 도 21의 트렐리스 디코더(2110)와 같은 트렐리스 디코더에 의해 생성되는 제1 연판정들 및 데이터의 비체계적 바이트나 비체계적 패킷에 관한 연판정들을 포함한다. 데이터의 체계적 및 비체계적 바이트는 바이트 코드 인코딩된 패킷으로부터 올 수 있다. 2/3 레이트 바이트 코드 디코더는 2 데이터 바이트를 디코딩하기 위해 3 바이트를 필요로 한다. 그러나, 원래의 연쇄 인코딩은 바람직하게는 하나의 비체계적 바이트를 제거함으로써 27 바이트로부터의 코드워드를 26 바이트로 줄이기 위해 1 바이트를 감소시킨다. 그 결과, 1 바이트가 인코딩 프로세스에서의 펑츄어링에 의해 제거된 바이트를 대체하기 위해 필요하다. 또한, 트렐리스 디코더는, 트렐리스 디코더로의 입력 스트림이 바이트를 포함하지 않았기 때문에, 데이터 스트림에서 펑츄어링된 바이트에 대한 어떠한 연판정들도 생성하지 않는다. 그 결과, 펑츄어링된 바이트의 값이 동일하게 가능하다는 것을 나타내는 연판정값이 삽입된다. 바이트 삽입 블록(2202)으로부터의 삽입된 연판정값을 포함한 제1 연판정들이 제1의 2/3 레이트 바이트 코드 디코더(2204)에 제공된다. 제1의 2/3 레이트 바이트 코드 디코더(2204)는 제1 연판정들을 이용하여 체계적 및 비체계적 패킷의 바이트를 디코딩하는 것에 기초하여 제2 연판정들을 생성한다. 연판정들의 생성은 예컨대 상기 수학식 (2) 및 (3)에 도시된 바이트 코딩된 패킷을 디벨롭하기 위해 사용된 b1 및 b2 성분의 값의 역(inverse)과 바이트 집합의 곱을 이용한다.

[0140] 제1의 2/3 레이트 바이트 코드 디코더로부터의 27 바이트 소프트가 펑츄어 블록(2206)에 제공된다. 27 바이트 소프트 출력은 제1의 2/3 레이트 바이트 코드 디코더에서의 디코딩 이후에 체계적 및 비체계적 바이트에 대한 갱신된 연판정값 집합을 나타낸다. 펑츄어 블록(2206)은 바이트 포맷을 트렐리스 디코더에 의해 원래 처리된 26

바이트 포맷으로 복구하기 위해 이전에 삽입된 연판정 바이트를 제거한다.

[0141] 체계적 바이트만을 나타내는 제1의 2/3 레이트 바이트 코드 디코더로부터의 18 바이트 소프트 출력은 디인터리버(2208)로 제공된다. 디인터리버(2208)는 2/3 레이트 바이트 코드 인코딩 프로세스에서 수행된 인터리빙을 역으로 하는 방식으로 18 바이트의 데이터를 디인터리빙한다. 디인터리버(2206)는 예컨대 도 7에서 행 710 및 720을 역으로 함으로써 인코더에서 인터리빙 맵을 정확히 역으로 한다.

[0142] 디인터리빙된 바이트는 제2의 2/3 레이트 바이트 코드 디코더(2210)로 제공된다. 제2의 2/3 레이트 바이트 코드 디코더(2210)는 디인터리빙된 연판정 체계적 바이트를 이용하여 상기의 것과 유사한 방식으로 연판정 바이트의 두개의 추가 출력을 생성한다. 18 바이트 소프트 출력이 인터리버(2212)로 제공된다. 18 바이트 소프트 출력은 제1의 2/3 레이트 바이트 코드 디코더(2204)에서의 디코딩으로부터의 체계적 및 비체계적 바이트 모두에 대한 갱신된 연판정값 집합을 나타낸다. 인터리버(2212)는 디인터리빙된 바이트를 제1의 2/3 레이트 바이트 코드 디코더에 의해 사용되는 바이트 포맷으로 다시 되돌려 놓기 위해 이를 다시 인터리빙한다. 인터리버(2212)는 도 6의 인터리버와 같은 인코더에서 사용되는 인터리버와 실질적으로 동일하고 다시 인터리빙된 18 바이트의 집합을 제1의 2/3 바이트 코드 디코더(2204)로 제공한다. 다시 인터리빙된 18 바이트의 집합은 제1의 2/3 레이트 바이트 코드 디코더(2204)에 의해 이루어진 연판정들을 개선하기 위해 사용된다.

[0143] 제2의 2/3 레이트 바이트 코드 디코더(2210)로부터의 12 바이트 출력은 12/26 레이트 바이트 코드 인코딩된 러기드 데이터 스트림에 대한 체계적 바이트 디코딩된 데이터 출력을 나타낸다. 제2의 2/3 레이트 바이트 코드 디코더(2210)에 의해 생성된 12 체계적 출력 바이트에 대한 연판정들이 최종적이거나 올바른 데이터 값들로서 최종적인 미리 정해진 한계값 내에 있으면, 제2의 2/3 레이트 바이트 코드 디코더(2210)는 연판정들을 이용하여 12 출력 바이트에 대한 경판정들을 생성하고 12 출력 바이트를 리드-솔로몬 디코더와 같은 추가 처리 블록으로 제공한다. 그러나, 제2의 2/3 레이트 바이트 코드 디코더에 의해 생성된 연판정들이 최종적이 아니라면, 이전의 반복 동안에 디벨롭되고 피드백된 소프트 정보를 이용하여 상기와 같이 추가 반복이 디벨롭된다. 이 추가적인 소프트 정보는 그 연속하는 디코더에 의해 각 소프트 디코더로 제공된다. 즉, 트렐리스 디코더는 펑츄어 블록(2206)을 통해 제공되는 제1의 2/3 레이트 바이트 코드 디코더(2204)로부터의 피드백을 이용하고, 제1의 2/3 레이트 바이트 코드 디코더(2204)는 인터리버(2212)를 통해 제공되는 제2의 2/3 레이트 바이트 코드 디코더(2210)로부터의 피드백을 이용한다. 이런 식으로 제2의 2/3 레이트 바이트 코드 디코더(2210)에 의해 생성된 연판정들이 충분히 수렴할 때까지 또는 소정 횟수의 반복이 수행될 때까지 반복이 계속된다. 상기한 바와 같이, 사용되는 터보 디코딩은 일반적으로 들어오는 데이터 레이트 보다 더 높은, 블록들 사이의 결정 데이터 전달에 관련된 반복 레이트를 이용한다.

[0144] 도 23을 참조하면, 연쇄 바이트 코드 디코더(2300)의 다른 실시예의 블록도가 도시된다. 연쇄 바이트 코드 디코더(2300)는 도 21에 도시된 것과 같은 터보 디코더 구성에서 동작하도록 유사하게 구성된다. 연쇄 바이트 코드 디코더(2300)는 또한 러기드 데이터 스트리에서 연쇄 바이트 코드 인코딩된 패킷을 디코딩하기 위해 3개의 성분 바이트 코드 디코더를 포함하는 반복 프로세스를 이용하는 터보 디코더로서 내부적으로 동작한다. 연쇄 바이트 코드 디코더(2300)는 레이트 12/52 블록 코드 인코딩된 신호 스트림을 디코딩하도록 적응되어 원래 인코딩된 52 바이트로부터 12 바이트의 데이터를 생성한다.

[0145] 52 바이트의 연판정값들을 나타내는 데이터 스트림이 패킷 삽입 블록(2302)으로 제공된다. 패킷 삽입 블록(2302)의 출력은 제1의 2/3 레이트 바이트 코드 디코더(2304)에 연결된다. 제1의 2/3 레이트 바이트 코드 디코더(2304)는 두개의 출력을 제공한다. 제1 출력은 펑츄어 블록(2306)에 연결되고, 펑츄어 블록의 출력은 도시되지 않았지만 인터리버를 통해 트렐리스 디코더에 피드백 입력으로 연결된다. 제1의 2/3 레이트 바이트 코드 디코더(2304)의 제2 출력은 제1 디인터리버(2308)에 연결된다. 제1 디인터리버(2308)의 출력은 역시 두개의 출력을 갖는 제2의 2/3 레이트 바이트 코드 디코더(2310)에 연결된다. 제1 출력은 제1 인터리버(2312)를 통해 제1의 2/3 레이트 바이트 코드 디코더(2304)에 피드백 입력으로 연결된다. 제2 출력은 제2 디인터리버(2314)에 연결된다. 제2 디인터리버(2314)의 출력은 역시 두개의 출력을 갖는 1/2 레이트 바이트 코드 디코더(2316)에 연결된다. 제1 출력은 제2 인터리버(2318)를 통해 제2의 2/3 레이트 바이트 코드 디코더(2310)에 피드백 입력으로 연결된다. 제2 출력은 리드-솔로몬 디코더와 같은 다른 처리 블록에 연결된다.

[0146] 트렐리스 디코더로부터의 제1 연판정들을 포함하는 52 바이트 입력이 패킷 삽입 블록(2302)에 제공된다. 패킷 삽입 블록(2302)은 52 바이트를 두 세트의 26 바이트로 분리한다. 이 분리는 도 9의 인코더(900)와 같은 인코더에 의해 바이트 코드 인코딩 중에 수행되는 분리와 매칭시키기 위해 수행된다. 패킷 삽입 블록(2302)은 두 세트의 27 바이트를 생성하기 위해 26 바이트의 각 세트에서 상술한 바와 같이 표시된 동일한 확률값을 갖는 연

판정 바이트를 삽입한다. 두 세트의 27 바이트는 추후 바이트 코드 디코딩 단계에서의 세트를 재조합하는 것을 허용하기 위해 처리에서 링크된 상태로 남아 있다. 제1의 2/3 레이트 바이트 코드 디코더(2304), 제1 디인터리버(2308), 및 제1 인터리버(2310)는 제1의 2/3 레이트 바이트 코드 디코더의 출력에서 두 세트의 27 연판정 바이트와 다음 두 세트의 18 연판정 바이트 사이의 링크를 처리하고 유지한다는 점을 제외하고 도 22에 도시된 것과 유사한 방식으로 동작한다. 펑츄어 블록(2306)은 두 세트의 27 소프트 출력 바이트로부터 이전에 삽입된 연판정 바이트를 제거하고 두 세트를 함께 연쇄한다. 이 연쇄는 트렐리스 디코더에 의해 원래 처리되는 52 바이트 포맷으로 바이트 포맷을 돌려 놓기 위해 필요하다. 펑츄어 블록(2306)에 제공되는 두개의 27 바이트 소프트 출력 세트는 제1의 2/3 레이트 바이트 코드 디코더에서의 디코딩 이후에 체계적 및 비체계적 바이트에 대한 갱신된 연판정값 세트를 나타낸다.

[0147] 제2의 2/3 레이트 바이트 코드 디코더(2310)는 상기한 대로 2 세트의 18 바이트를 처리하지만, 디코딩된 데이터의 체계적 바이트를 나타내는 2 세트의 12 바이트를 연쇄시켜 24 바이트 연판정 출력을 형성한다. 24 바이트 연판정 출력은 제2 디인터리버(2314)로 제공된다. 제2 디인터리버(2314)는 도 10에서 설명된 인코딩 프로세스의 일부로서 인터리버(1004)에서 수행된 인터리빙과 역의 방식으로 24 바이트의 데이터를 디인터리빙한다. 제2 디인터리버(2314)는 예컨대 도 11의 행 1110 및 1120을 역으로 함으로써 인코더에서의 인터리빙 맵을 역으로 한다.

[0148] 디인터리빙된 24 연판정 바이트는 1/2 레이트 바이트 코드 디코더(2316)에 제공된다. 1/2 레이트 바이트 코드 디코더(2316)는 디인터리빙된 연판정 체계적 바이트를 이용하여 상기의 것과 유사한 방식으로 연판정 바이트의 2개의 추가 출력을 생성한다. 24 바이트 소프트 출력은 제2 인터리버(2318)에 제공된다. 24 바이트 소프트 출력은 1/2 레이트 바이트 코드 디코더에서의 디코딩으로부터의 체계적 및 비체계적 바이트에 대한 갱신된 연판정값 세트를 나타낸다. 인터리버(2318)는 디인터리빙된 바이트를 제2의 2/3 레이트 바이트 코드 디코더(2310)에 의해 사용되는 포맷으로 되돌려 놓기 위해 이를 다시 인터리빙한다. 인터리버(2318)는 도 6의 인터리버(604)와 같은 인코더에서 사용되는 인터리버와 실질적으로 동일하고 다시 인터리빙된 24 바이트의 세트를 제2의 2/3 바이트 코드 디코더(2310)로 제공한다. 24 바이트의 다시 인터리빙된 세트는 제2의 2/3 레이트 바이트 코드 디코더(2310)에 의해 이루어진 연판정들을 개선하기 위해 사용된다.

[0149] 상기한 바와 같이, 1/2 레이트 바이트 코드 디코더(2316)의 출력에서 12 체계적 바이트에 관한 연판정들이 최종적이거나 올바른 데이터 값으로서 최종적인 것의 소정 한계값 내에 있다면, 1/2 레이트 바이트 코드 디코더(2316)는 12 출력 바이트에 관한 경판정을 생성하기 위해 연판정들을 이용하고 12 출력 바이트를 리드-솔로몬 디코더에서의 디코딩과 같은 추가 처리 블록으로 제공한다. 그러나, 1/2 레이트 바이트 코드 디코더(2316)에 의해 생성된 연판정들이 최종적이 아니면, 이전의 반복 동안에 디벨롭되고 피드백된 소프트 정보를 이용하여 추가 반복이 전개된다. 이 추가 소프트 정보는 그 연속하는 디코더에 의해 각 소프트 디코더에게 제공된다. 즉, 트렐리스 디코더는 펑츄어 블록(2306)을 통해 제공되는 제1의 2/3 레이트 바이트 코드 디코더(2304)로부터의 피드백을 이용하고, 제1의 2/3 레이트 바이트 코드 디코더(2304)는 제1 인터리버(2312)를 통해 제공되는 제2의 2/3 레이트 바이트 코드 디코더(2310)로부터의 피드백을 이용하고, 제2의 2/3 레이트 바이트 코드 디코더(2304)는 제2 인터리버(2318)을 통해 제공되는 1/2 레이트 바이트 코드 디코더(2316)로부터의 피드백을 이용한다. 이런 식으로 1/2 레이트 바이트 코드 디코더(2316)에 의해 생성된 연판정들이 충분히 수렴하거나 소정 횟수의 반복이 수행될 때까지 반복이 계속된다. 상기와 같이, 사용되는 터보 디코딩은 일반적으로 들어오는 데이터 레이트 보다 더 높은, 블록들 사이의 결정 데이터를 전달하는 것과 관련된 반복 레이트들을 이용한다.

[0150] 도 24를 참조하면, 연쇄 바이트 코드 디코더(2400)의 다른 실시예의 블록도가 도시되어 있다. 연쇄 바이트 코드 디코더(2400)는 도 21에 도시된 것과 같은 터보 디코더 구성에서 동작하도록 유사하게 구성된다. 연쇄 바이트 코드 디코더(2400)는 러기드 데이터 스트림에서의 바이트 코드 인코딩된 패킷과 병렬로 연결되어 동작하는, 두개의 성분 바이트 코드 디코더인, 2/3 레이트 디코더(2402) 및 1/2 레이트 디코더(2402)를 포함한다. 연쇄 바이트 코드 디코더(2400)는 레이트 17/26 바이트 코드 인코딩된 신호 스트림을 디코딩하도록 적응되어 원래 인코딩된 26 바이트로부터 17 바이트의 데이터를 생성한다.

[0151] 트렐리스 디코더로부터 26 바이트에 대한 연판정값들의 들어오는 데이터 스트림은 제1 그룹의 24 바이트와 제2 그룹의 2 바이트로 분리된다. 이 분리는 도 12의 인코더(1200)와 같은 인코더에 의해 바이트의 그룹의 조합을 위한 배열에 기초하여 수행되고 일반적으로 수신 장치에 알려져 있다. 이 분리는 도시되지 않았지만 신호 또는 패킷 멀티플렉서를 이용하여 수행될 수 있다. 이와 달리, 분리 및 그룹화는 예컨대 어느 바이트를 처리할지를 선택하기 위해 들어오는 바이트를 카운트함으로서 각 바이트 코드 인코더 내에서 수행될 수 있다. 제1 그룹의 24 연판정 바이트는 상기한 바와 같이 2/3 레이트 바이트 코드 디코더(2402)에 의해 디코딩된다. 제2 그룹의 2

연판정 바이트는 상기한 바와 같이 1/2 레이트 바이트 코드 디코더(2404)에 의해 유사하게 디코딩된다.

[0152] 2/3 레이트 바이트 코드 디코더(2402)에서의 디코딩으로부터 체계적 및 비체계적 바이트에 대한 갱신된 연판정 값 세트를 나타내는 24 바이트 소프트 출력과 1/2 레이트 바이트 코드 디코더(2404)에서의 디코딩으로부터 체계적 및 비체계적 바이트에 대한 갱신된 연판정값 세트를 나타내는 2 바이트 소프트 출력은 26 바이트 소프트 출력을 형성하기 위해 연쇄된다. 26 바이트 소프트 출력은 다음 터보 디코더 반복 동안에 바이트에 대한 소프트 값 판정들을 실질적으로 개선하기 위해 트렐리스 디코더로 제공된다.

[0153] 마찬가지로, 2/3 레이트 바이트 코드 디코더(2402)로부터의 체계적 바이트를 포함하는 16 바이트 소프트 출력과 1/2 레이트 바이트 코드 디코더(2404)로부터의 체계적 바이트를 포함하는 1 바이트 소프트 출력은 17 바이트 소프트 출력을 형성하기 위해 연쇄된다. 17 체계적 바이트에 대한 연판정들이 최종적이거나 올바른 데이터값으로서 최종적인 것의 미리 정해진 한계값 이내에 있으면, 2/3 레이트 바이트 코드 디코더(2402) 및 1/2 레이트 바이트 코드 디코더(2402)는 연판정들을 이용하여 17 출력 바이트에 관한 경판정들을 생성하고 17 출력 바이트를 리드-솔로몬 디코더와 같은 추가 처리 블록으로 제공한다.

[0154] 그러나, 17 출력 바이트에 대한 연판정들이 최종적이 아니면, 이전의 반복 동안에 디벨롭되고 피드백된 소프트 정보를 이용하여 추가 반복이 전개된다. 이 추가 소프트 정보는 그 연속하는 디코더에 의해 각 소프트 디코더로 제공된다. 즉, 트렐리스 디코더는 2/3 레이트 바이트 코드 디코더(2402) 및 1/2 레이트 바이트 코드 디코더(2404)의 연쇄 출력들로부터의 피드백을 이용한다. 이런 식으로 연판정들이 충분히 수렴하거나 소정 횟수의 반복이 수행될 때까지 반복이 계속된다. 상기와 같이, 사용된 터보 디코딩은 일반적으로 들어오는 데이터 레이트보다 더 높은, 블록들 사이의 결정 데이터를 전달하는 것에 관한 반복 레이트를 이용한다.

[0155] 도 20-24에서 설명된 바이트 코드 디코더는 도 13에 설명된 GF(256) SCBC에 의해 인코딩된 직렬 연쇄 블록 코드를 이용하여 인코딩된 데이터를 디코딩하도록 구성될 수 있음에 유의해야 한다.

[0156] 상기 바이트 코드 인코딩 및 디코딩의 배열을 이용하는 다양한 시스템은 기존 또는 레거시 방송 시스템의 적용의 확장을 허용한다. 첫째, 기존 수신기는 ATSC M/H를 이용하여 인코딩되는 추가적인 패킷들의 존재로부터 혜택을 얻을 수 있다. 보다 강건한 SCBC 인코딩 패킷 및 아프리오리 트래킹 패킷은 동적인 신호 환경 조건에서 트래킹을 개선하도록 트렐리스 디코더 및 이퀄라이저에 의해 처리될 수 있다. 둘째, 로버스트 또는 러기드 데이터를 생성하는 ATSC M/H 인코딩 데이터는 이동용, 휴대용 및 보행자용 장치에서의 수신 시스템이 레거시 A53 전송이 수신될 수 없는 신호 환경 하에서 로버스트 스트림을 수신할 수 있게 한다. 예컨대, 레이트 12/52에서의 ATSC M/H 인코딩은, 레거시 A53 수신에 대해 백색 잡음 한계값이 15dB 정도인 것에 비해, 3.5dB인 백색 잡음 한계값에서 신호 수신을 허용한다. ATSC M/H 패킷들을 생성하고 이 패킷들을 레거시 A53 데이터와 함께 주기적인 방식으로 전송함으로써 동작이 더욱 향상된다. 주기적 전송은 방송 재료의 비디오 및 오디오 전달을 허용하기 위해 중요하다. ATSC M/H 패킷들은 또한 하나 또는 그 이상의 전송 버스트로서 그룹화되어 전송될 수 있다. 버스트들로의 전송은 이동용, 휴대용, 또는 보행자용 장치에 의한 추후 사용을 위해 저장될 수 있는 데이터 콘텐츠 또는 콘텐츠의 전달을 위해 중요하다.

[0157] 특정 실시예들은 도면에서 예를 통해 제시되었고 상술되었지만, 본 실시예들에 대한 다양한 수정과 변형 형태도 가능하다. 그러나, 본 설명은 개시된 특정 형태에 한정되는 것이 아님은 물론이다. 그보다는, 이하의 청구범위에 의해 정의된 바와 같이 본 발명의 사상과 범위 내의 모든 수정, 균등물 및 변형을 포함하는 것이다.

도면의 간단한 설명

[0023] 도 1은 A53 표준에 따르는 신호를 전송하는 일반적인 송신 시스템의 블록도이다.

[0024] 도 2는 A53 표준에 따르는 신호를 수신하는 일반적인 수신기의 블록도이다.

[0025] 도 3은 본 발명의 인코더의 일 실시예의 블록도이다.

[0026] 도 4는 본 발명의 인코더의 다른 실시예의 블록도이다.

[0027] 도 5는 본 발명의 인코더의 또다른 실시예의 블록도이다.

[0028] 도 6은 본 발명의 연쇄(concatenated) 바이트 코드 인코더의 일 실시예의 블록도이다.

[0029] 도 7은 본 발명의 데이터 인터리빙의 맵을 도시하는 테이블이다.

[0030] 도 8은 본 발명의 연쇄 바이트 코드 인코더의 다른 실시예의 블록도이다.

[0031] 도 9는 본 발명의 연쇄 바이트 코드 인코더의 또다른 실시예의 블록도이다.

[0032] 도 10은 본 발명의 연쇄 바이트 코드 인코더의 또다른 실시예의 블록도이다.

[0033] 도 11은 본 발명의 데이터 인터리빙의 다른 맵을 도시하는 테이블이다.

[0034] 도 12는 본 발명의 연쇄 바이트 코드 인코더의 또다른 실시예의 블록도이다.

[0035] 도 13은 본 발명의 송신 장치에서 사용되는 인코더의 또다른 실시예의 블록도이다.

[0036] 도 14는 본 발명의 행 방향의 데이터를 도시하는 테이블이다.

[0037] 도 15는 본 발명의 열 방향의 데이터를 도시하는 테이블이다.

[0038] 도 16은 본 발명의 인코딩 프로세스의 일 실시예의 흐름도이다.

[0039] 도 17은 본 발명의 인코딩 프로세스의 다른 실시예의 흐름도이다.

[0040] 도 18은 본 발명의 심볼들에 대한 비트들의 매핑을 도시하는 테이블이다.

[0041] 도 19는 본 발명의 인터리버에서 바이트들의 매핑을 도시하는 테이블이다.

[0042] 도 20은 본 발명의 디코더의 일 실시예의 블록도이다.

[0043] 도 21은 본 발명의 디코더의 다른 실시예의 블록도이다.

[0044] 도 22는 본 발명의 연쇄 바이트 코드 디코더의 일 실시예의 블록도이다.

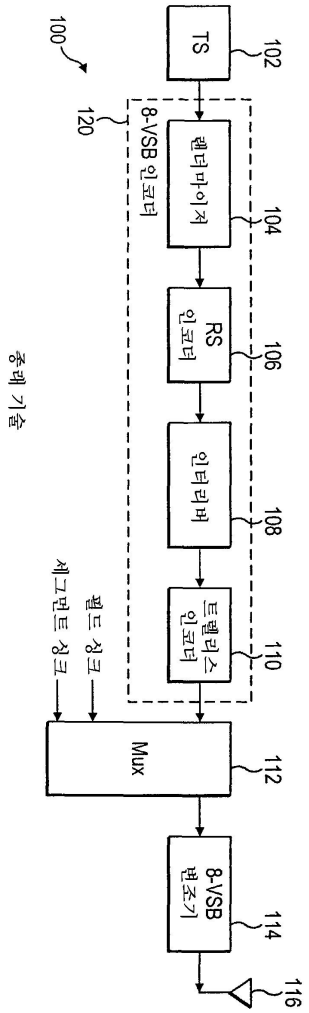
[0045] 도 23은 본 발명의 연쇄 바이트 코드 디코더의 다른 실시예의 블록도이다.

[0046] 도 24는 본 발명의 연쇄 바이트 코드 디코더의 또다른 실시예의 블록도이다.

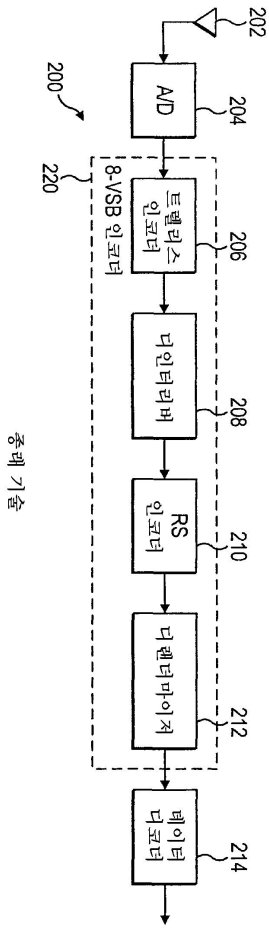
[0047] 본 발명의 특징 및 이점은 예를 드는 식으로 주어진 이하의 설명으로부터 보다 명백해질 수 있다.

도면

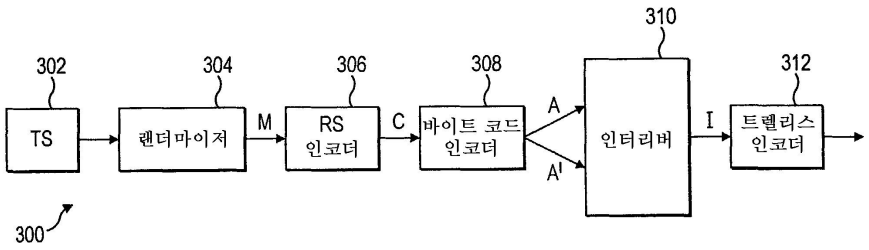
도면1


종래 기술
100
TS
102
8-VSB 인코더
120
랜더마이저
104
RS 인코더
106
인터리버
108
트렐리스 인코더
110
필드 싱크
세그먼트 싱크
Mux
112
8-VSB 변조기
114
116

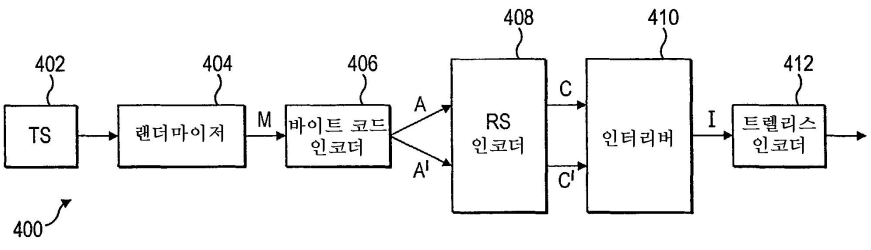
도면2


202
200
A/D
204
220
8-VSB 인코더
트렐리스 인코더
206
디인터리버
208
RS 인코더
210
디랜더마이저
212
베이터 디코더
214
종래 기술

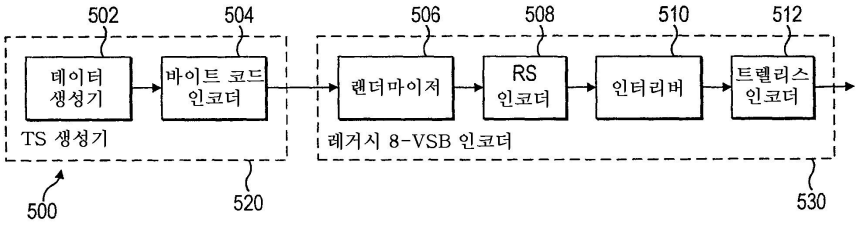
도면3


310
302
304
306
308
312
TS
랜더마이저
M
RS 인코더
C
바이트 코드 인코더
A
A'
인터리버
I
트렐리스 인코더
300

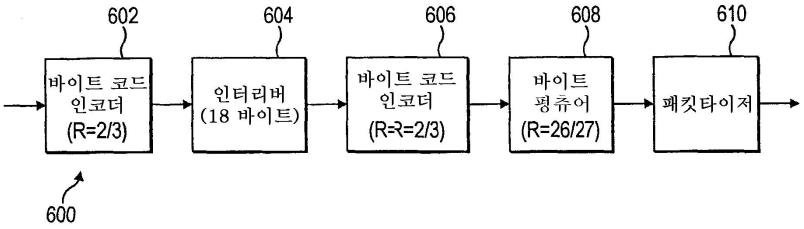
도면4


408
410
402
404
406
412
TS
랜더마이저
M
바이트 코드 인코더
A
A'
RS 인코더
C
C'
인터리버
I
트렐리스 인코더
400

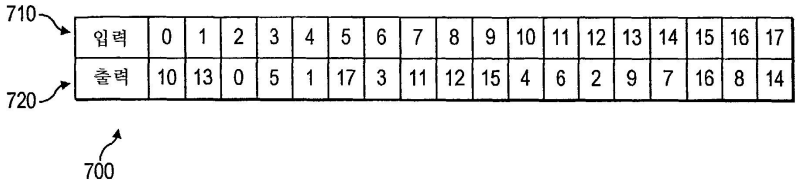
도면5



도면6

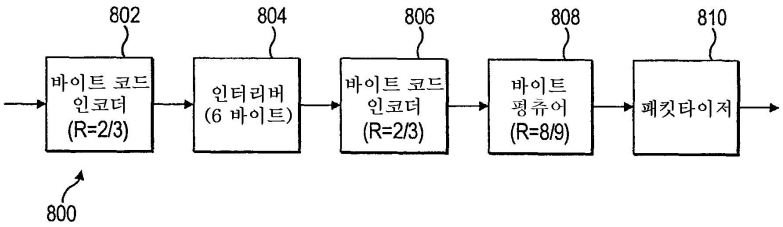


도면7

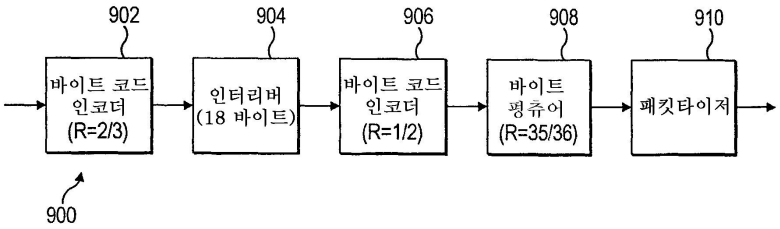


입력	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
출력	10	13	0	5	1	17	3	11	12	15	4	6	2	9	7	16	8	14

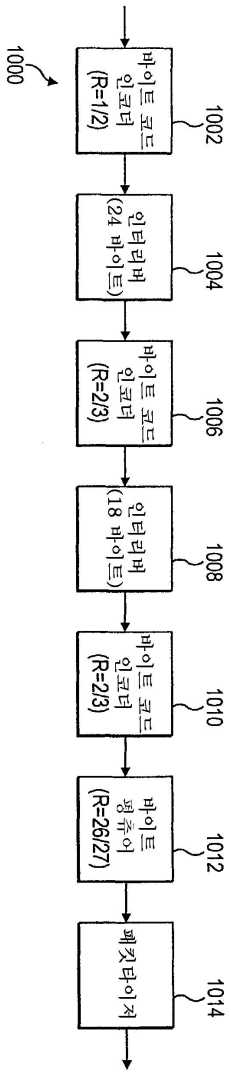
도면8



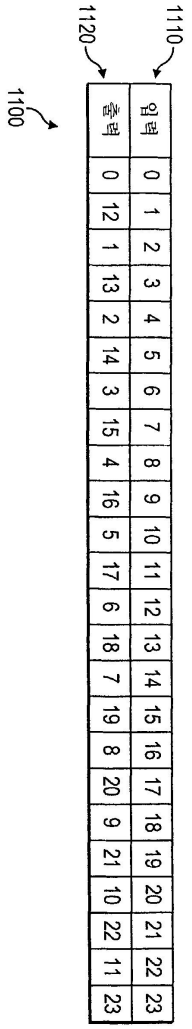
도면9


902
904
906
908
910
바이트 코드 인코더 (R=2/3)
인터리버 (18 바이트)
바이트 코드 인코더 (R=1/2)
바이트 펑츄어 (R=35/36)
패킷타이저
900

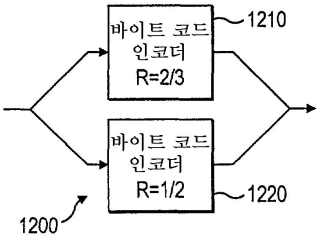
도면10


1000
바이트 코드 인코더 (R=1/2)
1002
인터리버 (24 바이트)
1004
바이트 코드 인코더 (R=2/3)
1006
인터리버 (18 바이트)
1008
바이트 코드 인코더 (R=2/3)
1010
바이트 펑츄어 (R=26/27)
1012
패킷타이저
1014

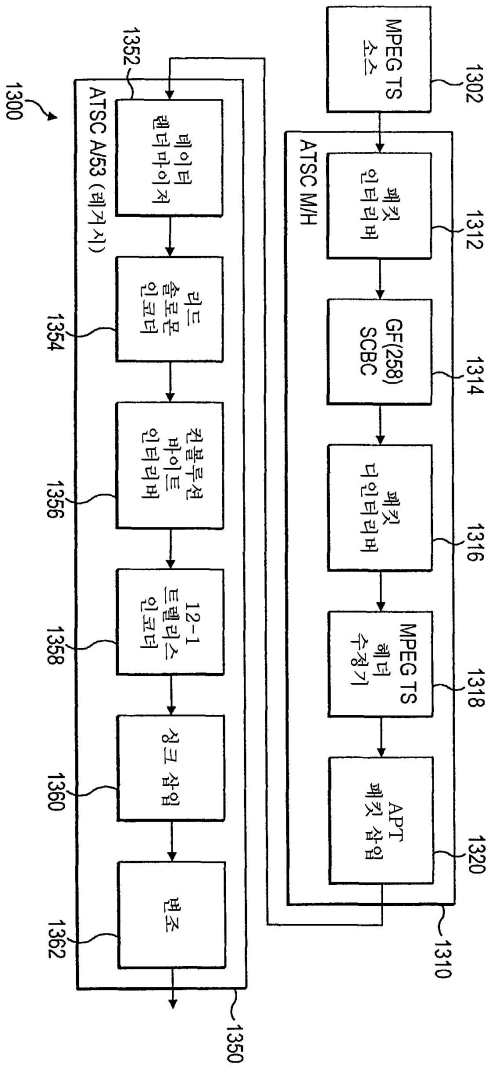
도면11


1120
1110
1100
출력
입력
0 0
12 1
1 2
13 3
2 4
14 5
3 6
15 7
4 8
16 9
5 10
17 11
6 12
18 13
7 14
19 15
8 16
20 17
9 18
21 19
10 20
22 21
11 22
23 23

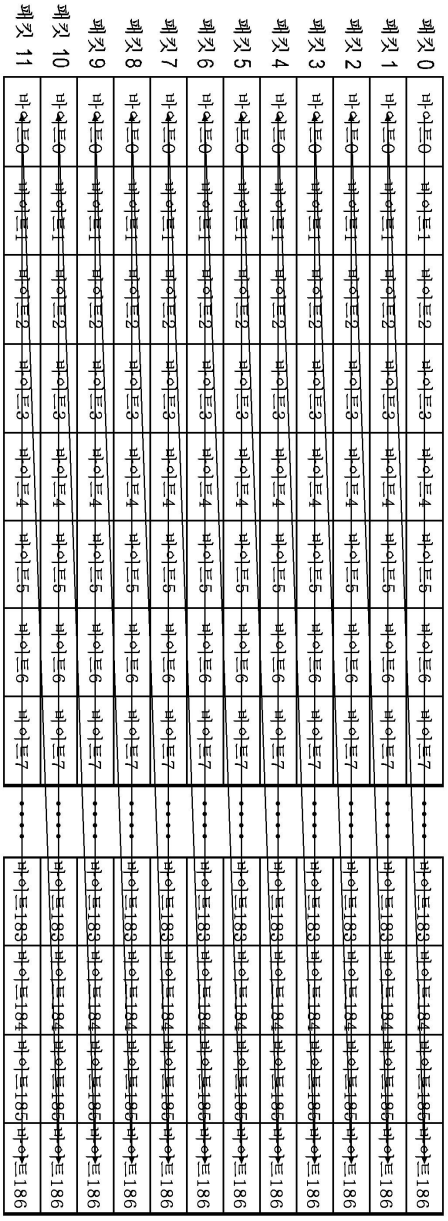
도면12


바이트 코드
인코더
R=2/3
1210
바이트 코드
인코더
R=1/2
1220
1200

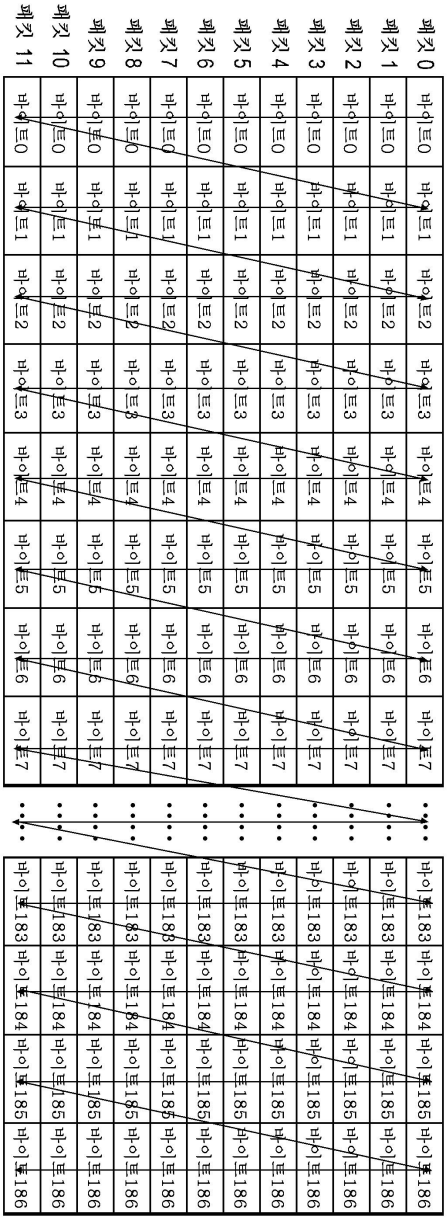
도면13


MPEG TS 소스
1302
ATSC M/H
패킷 인터리버
1312
GF(258) SCBC
1314
패킷 디인터리버
1316
MPEG TS 헤더 수정기
1318
APT 패킷 삽입
1320
1310
1300
ATSC A/53 (레거시)
1352
데이터 랜더마이저
리드 솔로몬 인코더
1354
컨볼루션 바이트 인터리버
1356
12-1 트렐리스 인코더
1358
싱크 삽입
1360
변조
1362
1350

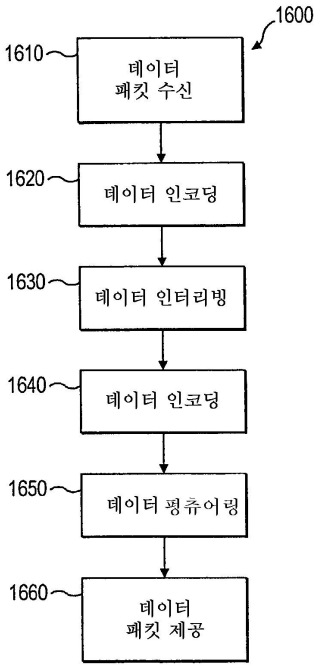
도면14


워드 0
워드 1
워드 2
워드 3
워드 4
워드 5
워드 6
워드 7
워드 8
워드 9
워드 10
워드 11
비트0
비트1
비트2
비트3
비트4
비트5
비트6
비트7
비트8
비트9
비트10
비트11
비트12
비트13
비트14
비트15
비트16
비트17
비트18
비트19
비트20
비트21
비트22
비트23
비트24
비트25
비트26
비트27
비트28
비트29
비트30
비트31
비트32
비트33
비트34
비트35
비트36
비트37
비트38
비트39
비트40
비트41
비트42
비트43
비트44
비트45
비트46
비트47
비트48
비트49
비트50
비트51
비트52
비트53
비트54
비트55
비트56
비트57
비트58
비트59
비트60
비트61
비트62
비트63
비트64
비트65
비트66
비트67
비트68
비트69
비트70
비트71
비트72
비트73
비트74
비트75
비트76
비트77
비트78
비트79
비트80
비트81
비트82
비트83
비트84
비트85
비트86

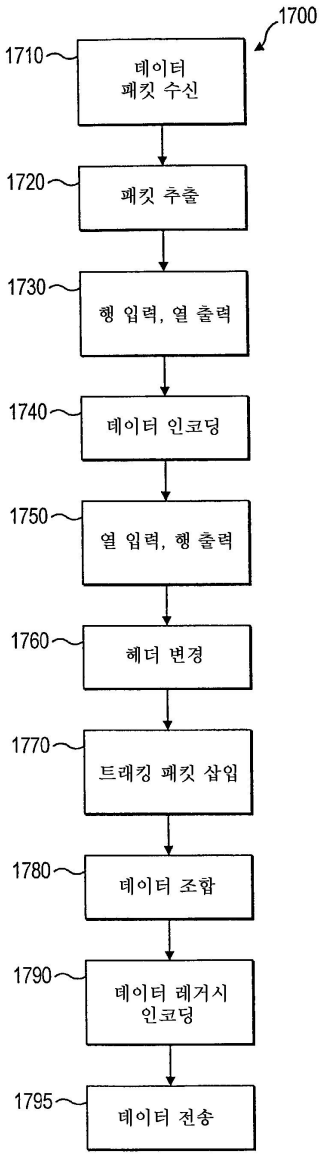
도면15


패킷 0
패킷 1
패킷 2
패킷 3
패킷 4
패킷 5
패킷 6
패킷 7
패킷 8
패킷 9
패킷 10
패킷 11
바이트0
바이트1
바이트2
바이트3
바이트4
바이트5
바이트6
바이트7
바이트183
바이트184
바이트185
바이트186

도면16


1600
1610
데이터
패킷 수신
1620
데이터 인코딩
1630
데이터 인터리빙
1640
데이터 인코딩
1650
데이터 펑츄어링
1660
데이터
패킷 제공

도면17

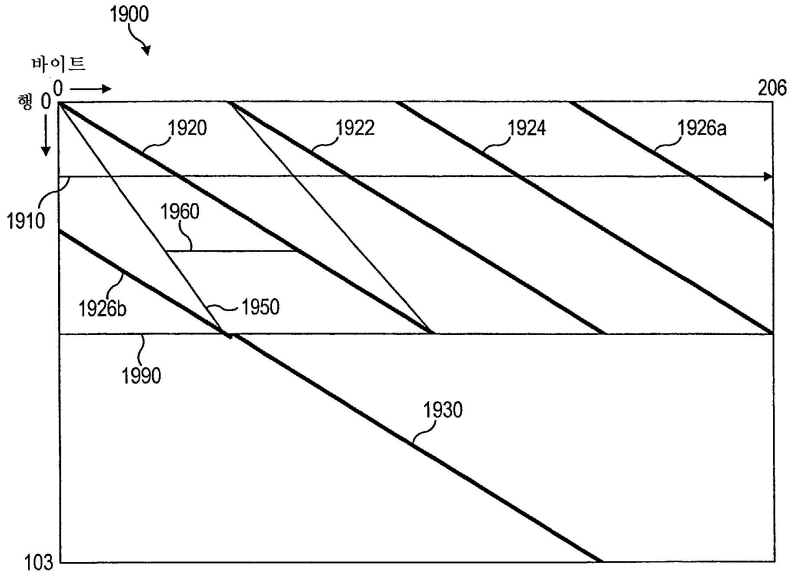


도면18

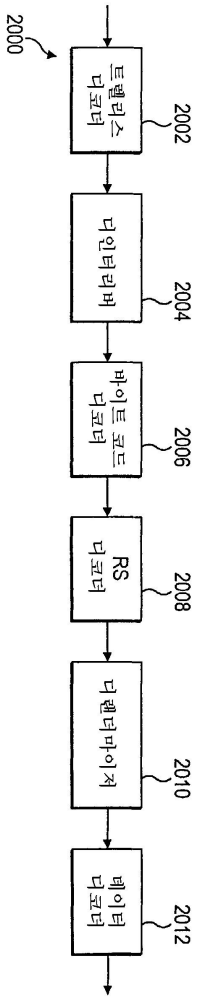
1800

심볼	Z2	Z1	진폭
0	0	0	+3
1	0	1	+1
2	1	0	-1
3	1	1	-3

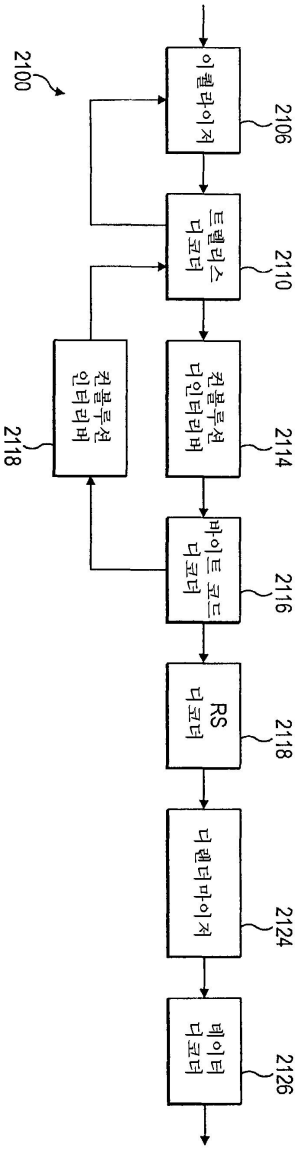
도면19


1900
바이트
0
206
행 0
1920
1922
1924
1926a
1910
1960
1926b
1950
1990
1930
103

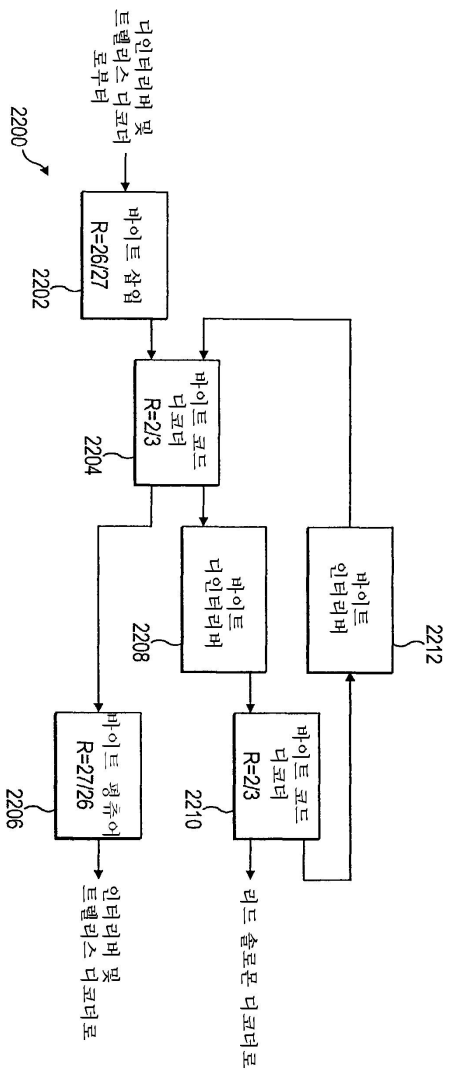
도면20


2000
트렐리스 디코더
2002
디인터리버
2004
바이트 코드 디코더
2006
RS 디코더
2008
디랜더마이저
2010
베이터 디코더
2012

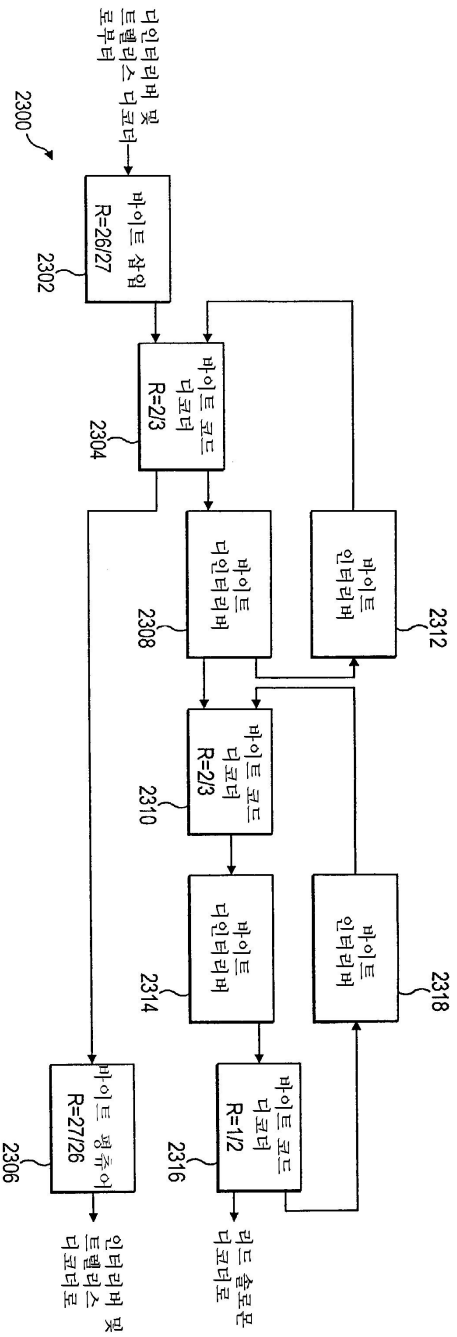
도면21


2100
이퀄라이저
2106
트렐리스 디코더
2110
컨볼루션 디인터리버
2114
바이트 코드 디코더
2116
RS 디코더
2118
디랜더마이저
2124
데이터 디코더
2126
컨볼루션 인터리버
2118

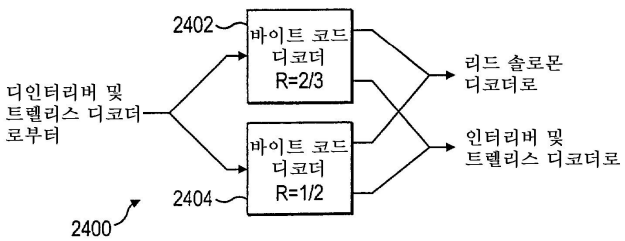
도면22


디인터리버 및
트렐리스 디코더
로부터
2200
바이트 삽입
R=26/27
2202
바이트 코드
디코더
R=2/3
2204
바이트
디인터리버
2208
바이트
인터리버
2212
바이트 펑쳐
R=27/26
2206
바이트 코드
디코더
R=2/3
2210
인터리버 및
트렐리스 디코더로
리드 솔로몬 디코더로

도면23


디인터리버 및 트렐리스 디코더로부터
2300
바이트 삽입 R=26/27
2302
바이트 코드 디코더 R=2/3
2304
바이트 디인터리버
2308
바이트 인터리버
2312
바이트 코드 디코더 R=2/3
2310
바이트 디인터리버
2314
바이트 인터리버
2318
바이트 펑쳐 R=27/26
2306
바이트 코드 디코더 R=1/2
2316
인터리버 및 트렐리스 디코더로
리드 솔로몬 디코더로

도면24


2402
바이트 코드 디코더 R=2/3
디인터리버 및 트렐리스 디코더로부터
바이트 코드 디코더 R=1/2
2404
2400
리드 솔로몬 디코더로
인터리버 및 트렐리스 디코더로