(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2011-0082796
(43) 공개일자 2011년07월20일

(51) Int. Cl.
H04L 29/04 (2006.01)
(21) 출원번호 10-2010-0002686
(22) 출원일자 2010년01월12일
심사청구일자 2010년01월12일

(71) 출원인
연세대학교 산학협력단
서울 서대문구 신촌동 134 연세대학교
(72) 발명자
노원우
서울특별시 종로구 무악동 인왕산 I-PARK아파트 113동 804호
김선우
서울특별시 중구 신당4동 약수하이츠아파트 114-1002
(74) 대리인
송윤호, 오세준, 권혁수

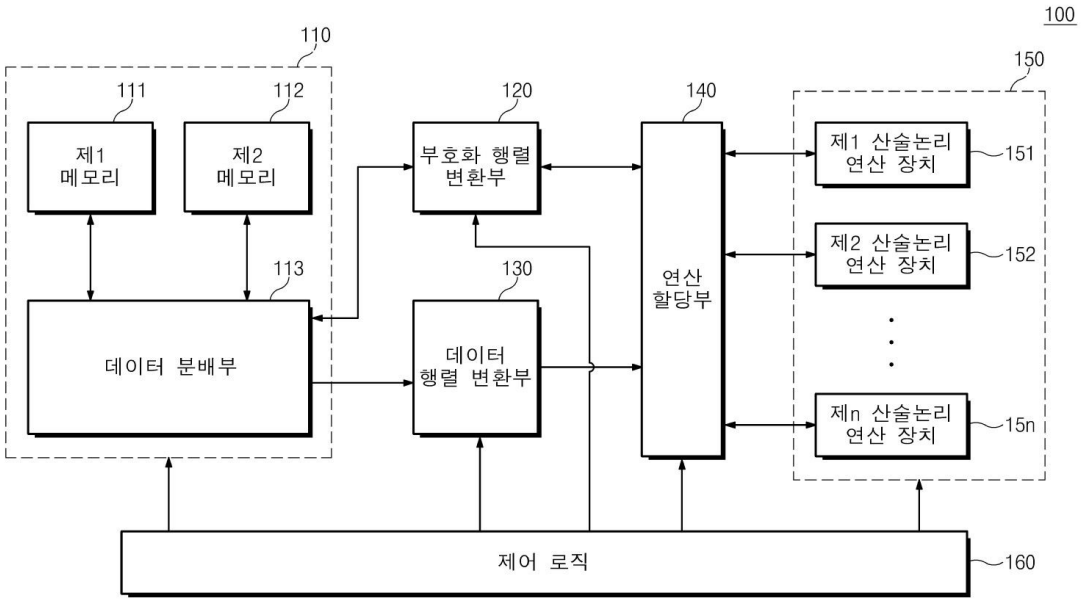
전체 청구항 수 : 총 20 항

(54) 네트워크 코딩의 데이터 복원 장치 및 그것의 데이터 복원 방법

(57) 요 약

본 발명은 네트워크 코딩의 데이터 복원 장치 및 그것의 데이터 복원 방법에 관한 것이다. 본 발명의 기술적 사상의 실시 예에 따른 네트워크 코딩의 데이터 복원 장치는 데이터 패킷들을 전달받아, 상기 데이터 패킷들의 부호화 벡터들을 부호화 행렬로 저장하는 패킷 저장부 및 상기 부호화 행렬을 전달받아, 상기 부호화 행렬의 역행렬을 연산하는 연산부를 포함하며, 상기 연산부는 행 단위로 상기 부호화 행렬의 역행렬을 연산하는 복수의 연산 기들을 포함한다. 본 발명의 기술적 사상의 실시 예에 따른 네트워크 코딩의 데이터 복원 장치 및 그것의 데이터 복원 방법은 원본 데이터를 빠르게 복원할 수 있다.

대 표 도



특허청구의 범위

청구항 1

데이터 패킷들을 전달받아, 상기 데이터 패킷들의 부호화 벡터들을 부호화 행렬로 저장하는 패킷 저장부; 및

상기 부호화 행렬을 전달받아, 상기 부호화 행렬의 역행렬을 연산하는 연산부를 포함하며,

상기 연산부는 행 단위로 상기 부호화 행렬의 역행렬을 연산하는 네트워크 코딩의 데이터 복원 장치.

청구항 2

제 1 항에 있어서,

상기 연산부는 상기 부호화 행렬 중 선택된 행의 원소들을 각각 전달받아 상기 부호화 행렬의 역행렬을 병렬적으로 연산하는 복수의 연산기들을 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 3

제 2 항에 있어서,

상기 복수의 연산기들 각각은 가우스-조던 소거법(Gaussian Jordan elimination)에 의하여 상기 부호화 행렬의 역행렬을 연산하는 네트워크 코딩의 데이터 복원 장치.

청구항 4

제 3 항에 있어서,

상기 복수의 연산기들 각각은 상기 부호화 행렬 중 선택된 타겟 행의 원소들을 각각 전달받고, 상기 부호화 행렬 중 선택된 피봇 행의 원소들을 각각 전달받아 상기 부호화 행렬의 역행렬을 연산하는 네트워크 코딩의 데이터 복원 장치.

청구항 5

제 2 항에 있어서,

상기 패킷 저장부에 연결되며, 상기 패킷 저장부에 저장된 부호화 행렬을 저장하는 제 1 레지스터를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 6

제 5 항에 있어서,

상기 제 1 레지스터에 연결되며, 상기 부호화 행렬의 선택된 행을 저장하는 제 2 레지스터를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 7

제 6 항에 있어서,

상기 부호화 행렬에 대응하는 단위 행렬을 저장하는 제 3 레지스터를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 8

제 7 항에 있어서,

상기 제 3 레지스터에 연결되며, 상기 단위 행렬의 선택된 행을 저장하는 제 4 레지스터를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 9

제 8 항에 있어서,

상기 제 2 레지스터 및 상기 제 4 레지스터에 연결되며, 상기 제 2 레지스터에 저장된 상기 부호화 행렬 중 선택된 행의 원소들 및 상기 제 4 레지스터에 저장된 상기 단위 행렬 중 선택된 행의 원소들을 상기 복수의 연산기들에 각각 분배하는 할당부를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 10

제 8 항에 있어서,

상기 패킷 저장부는 상기 부호화 행렬을 저장하는 제 1 메모리 및 상기 데이터 패킷들의 부호화 데이터 행렬을 저장하는 제 2 메모리를 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 11

제 10 항에 있어서,

상기 부호화 데이터 행렬 중 선택된 열을 저장하는 제 5 레지스터를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 12

제 10 항에 있어서,

상기 부호화 행렬의 역행렬을 저장하는 제 6 레지스터를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 13

제 12 항에 있어서,

상기 제 5 레지스터 및 상기 제 6 레지스터에 연결되며, 상기 제 5 레지스터에 저장된 상기 부호화 데이터 행렬 중 선택된 열의 원소들 및 상기 제 6 레지스터에 저장된 상기 부호화 행렬의 역행렬 중 선택된 행의 원소들을 상기 복수의 연산기들에 각각 분배하는 할당부를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 14

제 13 항에 있어서,

상기 복수의 연산기들은 각각은 상기 부호화 데이터 행렬 중 선택된 열의 원소들 및 상기 부호화 행렬의 역행렬 중 선택된 행의 원소들의 곱셈에 기초하여, 상기 부호화 데이터 행렬의 원본 데이터를 복원하는 네트워크 코딩의 데이터 복원 장치.

청구항 15

제 2 항에 있어서,

상기 복수의 연산기들 각각은

상기 부호화 행렬의 선택된 제 1 행의 원소를 전달받아, 상기 선택된 제 1 행의 원소에 대응하는 제 1 로그값으로 변환하는 제 1 변환부; 및

상기 부호화 행렬의 선택된 제 2 행의 원소를 전달받아, 상기 선택된 제 2 행의 원소에 대응하는 제 2 로그값으로 변환하는 제 2 변환부를 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 16

제 15 항에 있어서,

상기 복수의 연산기들 각각은

상기 제 1 변환부 및 상기 제 2 변환부에 연결되며, 상기 제 1 로그값 및 상기 제 2 로그값에 대한 덧셈 연산을 수행하는 덧셈기를 더 포함하는 네트워크 코딩의 데이터 복원 장치.

청구항 17

제 16 항에 있어서,

상기 덧셈기에 연결되며, 상기 제 1 로그값 및 상기 제 2 로그값에 대한 덧셈 연산 결과에 대한 역 로그 변환을 수행하여 원본 데이터를 복원하는 제 3 변환부를 더 포함하는 네트워크 코딩이 데이터 복원 장치.

청구항 18

데이터 패킷을 전달받아, 상기 데이터 패킷들의 부호화 벡터들 및 상기 데이터 패킷들의 부호화 블록들을 각각 부호화 행렬 및 부호화 데이터 행렬로 저장하는 단계;

상기 부호화 행렬을 전달받아, 상기 부호화 행렬의 역행렬을 연산하는 단계; 및

상기 부호화 데이터 행렬 및 상기 부호화 행렬의 역행렬을 전달받아, 원본 데이터를 복원하는 단계를 포함하되,

상기 부호화 행렬의 역행렬을 연산하는 단계는 상기 부호화 행렬을 행 단위로 연산하는 네트워크 코딩의 데이터 복원 방법.

청구항 19

제 18 항에 있어서,

상기 부호화 행렬의 역행렬을 연산하는 단계는 상기 부호화 행렬 중 선택된 타겟 행의 원소들과 상기 부호화 행렬 중 선택된 피봇 행의 원소들을 각각 병렬적으로 연산하는 네트워크 코딩의 데이터 복원 방법.

청구항 20

제 18 항에 있어서,

상기 원본 데이터를 복원하는 단계는 상기 부호화 데이터 행렬 중 선택된 열 및 상기 부호화 행렬의 역행렬 중 선택된 행의 곱셈에 기초하여, 상기 원본 데이터를 복원하는 네트워크 코딩의 데이터 복원 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 네트워크 코딩에 관한 것으로, 좀더 상세하게는 네트워크 코딩의 데이터 복원 장치 및 그것의 데이터 복원 방법에 관한 것이다.

배 경 기 술

[0002] 네트워크 코딩(network coding) 기법을 사용하는 통신망은 일반적인 통신망과 달리 중간 경유 노드 또는 라우터에서 서로 다른 패킷들을 혼합한다. 일반적인 통신망의 송신단에서 생성된 패킷은 중간 경유 노드 또는 라우터에서 변경되지 않고 수신단까지 전달된다. 그러나 네트워크 코딩 기법을 사용하는 통신망은 중간 경유 노드 또는 라우터에서 서로 다른 패킷들의 혼합을 허용하거나, 패킷의 내용의 변경을 허용한다.

[0003] 한편, 네트워크 코딩 기법을 사용하는 통신망의 경우에, 원본 데이터는 직접 전송되는 대신에 부호화된 데이터 조각들로 전송된다. 예를 들어, 이러한 부호화된 데이터 조각은 데이터 패킷이라 칭해질 수 있다. 네트워크 코딩의 수신단은 데이터 패킷들을 전달받아 원본 데이터를 복원한다.

발명의 내용

해결하려는 과제

[0004] 본 발명의 목적은 원본 데이터를 빠르게 복원할 수 있는 네트워크 코딩의 데이터 복원 장치 및 그것의 데이터 복원 방법을 제공하는 데 있다.

과제의 해결 수단

[0005] 본 발명의 기술적 사상의 실시 예에 따른 네트워크 코딩의 데이터 복원 장치는 데이터 패킷들을 전달받아, 상기

데이터 패킷들의 부호화 벡터들을 부호화 행렬로 저장하는 패킷 저장부 및 상기 부호화 행렬을 전달받아, 상기 부호화 행렬의 역행렬을 연산하는 연산부를 포함하며, 상기 연산부는 행 단위로 상기 부호화 행렬의 역행렬을 연산한다.

[0006] 실시 예로서, 상기 연산부는 상기 부호화 행렬 중 선택된 행의 원소들을 각각 전달받아 상기 부호화 행렬의 역행렬을 병렬적으로 연산하는 복수의 연산기들을 포함한다.

[0007] 실시 예로서, 상기 복수의 연산기들 각각은 가우스-조던 소거법(Gaussian Jordan elimination)에 의하여 상기 부호화 행렬의 역행렬을 연산한다.

[0008] 실시 예로서, 상기 복수의 연산기들 각각은 상기 부호화 행렬 중 선택된 타겟 행의 원소들을 각각 전달받고, 상기 부호화 행렬 중 선택된 피봇 행의 원소들을 각각 전달받아 상기 부호화 행렬의 역행렬을 연산한다.

[0009] 실시 예로서, 상기 패킷 저장부에 연결되며, 상기 패킷 저장부에 저장된 부호화 행렬을 저장하는 제 1 레지스터를 더 포함한다.

[0010] 실시 예로서, 상기 제 1 레지스터에 연결되며, 상기 부호화 행렬의 선택된 행을 저장하는 제 2 레지스터를 더 포함한다.

[0011] 실시 예로서, 상기 부호화 행렬에 대응하는 단위 행렬을 저장하는 제 3 레지스터를 더 포함한다.

[0012] 실시 예로서, 상기 제 3 레지스터에 연결되며, 상기 단위 행렬의 선택된 행을 저장하는 제 4 레지스터를 더 포함한다.

[0013] 실시 예로서, 상기 제 2 레지스터 및 상기 제 4 레지스터에 연결되며, 상기 제 2 레지스터에 저장된 상기 부호화 행렬 중 선택된 행의 원소들 및 상기 제 4 레지스터에 저장된 상기 단위 행렬 중 선택된 행의 원소들을 상기 복수의 연산기들에 각각 분배하는 할당부를 더 포함한다.

[0014] 실시 예로서, 상기 패킷 저장부는 상기 부호화 행렬을 저장하는 제 1 메모리 및 상기 데이터 패킷들의 부호화 데이터 행렬을 저장하는 제 2 메모리를 포함한다.

[0015] 실시 예로서, 상기 부호화 데이터 행렬 중 선택된 열을 저장하는 제 5 레지스터를 더 포함한다.

[0016] 실시 예로서, 상기 부호화 행렬의 역행렬을 저장하는 제 6 레지스터를 더 포함한다.

[0017] 실시 예로서, 상기 제 5 레지스터 및 상기 제 6 레지스터에 연결되며, 상기 제 5 레지스터에 저장된 상기 부호화 데이터 행렬 중 선택된 열의 원소들 및 상기 제 6 레지스터에 저장된 상기 부호화 행렬의 역행렬 중 선택된 행의 원소들을 상기 복수의 연산기들에 각각 분배하는 할당부를 더 포함한다.

[0018] 실시 예로서, 상기 복수의 연산기들은 각각은 상기 부호화 데이터 행렬 중 선택된 열의 원소들 및 상기 부호화 행렬의 역행렬 중 선택된 행의 원소들의 곱셈에 기초하여, 상기 부호화 데이터 행렬의 원본 데이터를 복원한다.

[0019] 실시 예로서, 상기 복수의 연산기들 각각은 상기 부호화 행렬의 선택된 제 1 행의 원소를 전달받아, 상기 선택된 제 1 행의 원소에 대응하는 제 1 로그값으로 변환하는 제 1 변환부 및 상기 부호화 행렬의 선택된 제 2 행의 원소를 전달받아, 상기 선택된 제 2 행의 원소에 대응하는 제 2 로그값으로 변환하는 제 2 변환부를 포함한다.

[0020] 실시 예로서, 상기 복수의 연산기들 각각은 상기 제 1 변환부 및 상기 제 2 변환부에 연결되며, 상기 제 1 로그값 및 상기 제 2 로그값에 대한 덧셈 연산을 수행하는 덧셈기를 더 포함한다.

[0021] 실시 예로서, 상기 덧셈기에 연결되며, 상기 제 1 로그값 및 상기 제 2 로그값에 대한 덧셈 연산 결과에 대한 역 로그 변환을 수행하여 원본 데이터를 복원하는 제 3 변환부를 더 포함한다.

[0022] 본 발명의 기술적 사상의 실시 예에 따른 네트워크 코딩의 데이터 복원 방법은 데이터 패킷을 전달받아, 상기 데이터 패킷들의 부호화 벡터들 및 상기 데이터 패킷들의 부호화 블록들을 각각 부호화 행렬 및 부호화 데이터 행렬로 저장하는 단계, 상기 부호화 행렬을 전달받아, 상기 부호화 행렬의 역행렬을 연산하는 단계; 및 상기 부호화 데이터 행렬 및 상기 부호화 행렬의 역행렬을 전달받아, 원본 데이터를 복원하는 단계를 포함하되, 상기 부호화 행렬의 역행렬을 연산하는 단계는 상기 부호화 행렬을 행 단위로 연산한다.

[0023] 실시 예로서, 상기 부호화 행렬의 역행렬을 연산하는 단계는 상기 부호화 행렬 중 선택된 타겟 행의 원소들과 상기 부호화 행렬 중 선택된 피봇 행의 원소들을 각각 병렬적으로 연산한다.

[0024] 실시 예로서, 상기 원본 데이터를 복원하는 단계는 상기 부호화 데이터 행렬 중 선택된 열 및 상기 부호화 행렬

의 역행렬 중 선택된 행의 곱셈에 기초하여, 상기 원본 데이터를 복원한다.

발명의 효과

[0025] 본 발명의 기술적 사상의 실시 예에 따른 네트워크 코딩의 데이터 복원 장치 및 그것의 데이터 복원 방법은 원본 데이터를 빠르게 복원할 수 있다.

도면의 간단한 설명

[0026] 도 1은 본 발명의 기술적 사상의 실시 예에 따른 통신망에서의 네트워크 코딩 기법의 적용 예를 보여준다.

도 2는 본 발명의 기술적 사상의 실시 예에 따른 데이터 패킷들을 보여주는 블록도이다.

도 3은 본 발명의 기술적 사상의 실시 예에 따른 데이터 복원 장치를 보여주는 블록도이다.

도 4는 도 2의 데이터 복원 장치(100)의 동작을 설명하기 위한 순서도이다.

도 5는 본 발명의 기술적 사상의 실시 예에 따른 도 3의 부호화 행렬 변환부(120)를 좀더 상세하게 보여주는 블록도이다.

도 6은 도 5의 부호화 행렬 변환부(120)의 동작을 좀더 상세하게 설명하기 위한 순서도이다.

도 7은 본 발명의 기술적 사상의 실시 예에 따른 도 3의 데이터 행렬 변환부(130)를 좀더 상세하게 보여주는 블록도이다.

도 8은 도 7의 데이터 행렬 변환부(130)의 동작을 좀더 자세하게 설명하기 위한 순서도이다.

도 9는 본 발명의 기술적 사상의 실시 예에 따른 도 3의 산술 논리 연산부(150)의 구조를 보여주는 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0027] 이하, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명의 기술적 사상을 용이하게 실시할 수 있을 정도로 상세히 설명하기 위하여, 본 발명의 기술적 사상의 실시 예에 따른 도면을 참조하여 설명한다.

[0028] 도 1은 본 발명의 기술적 사상의 실시 예에 따른 통신망에서의 네트워크 코딩 기법의 적용 예를 보여준다. 도 1에서는 통신망의 예로서 버터 플라이 네트워크(Butterfly network)가 도시되어 있다. 도 1을 참조하면, 통신망(10)은 서버(11), 제 1 내지 제 4 노드들(12~15), 제 1 및 제 2 컴퓨터들(16, 17)을 포함한다.

[0029] 네트워크 코딩 기법은 각 노드에서 들어오는 정보들을 적절한 결합을 통하여 다른 노드로 전달한다. 예를 들어 도 1을 참조하면, 제 3 노드(14)는 제 1 노드(12)로부터 데이터 'a'를 전달받는다. 제 3 노드(14)는 제 2 노드(13)로부터 데이터 'b'를 전달받는다. 네트워크 코딩 기법이 적용되는 경우, 제 3 노드(14)는 XOR 연산을 통하여 'a $\oplus$ b' 의 데이터를 제 4 노드(15)에 전달한다.

[0030] 제 1 컴퓨터(16)는 제 1 노드(12)로부터 데이터 'a'를 전달받는다. 제 1 컴퓨터(16)는 제 4 노드(15)로부터 데이터 'a $\oplus$ b'를 전달받는다. 제 1 컴퓨터(16)는 수신된 데이터들을 XOR 연산하여 데이터 'a' 및 데이터 'b'를 얻을 수 있다. 또한, 제 2 컴퓨터(17)는 제 2 노드(13)로부터 데이터 'b'를 전달받는다. 제 2 컴퓨터(17)는 제 4 노드(15)로부터 데이터 'a $\oplus$ b'를 전달받는다. 제 2 컴퓨터(17)는 수신된 데이터들을 XOR 연산하여 데이터 'a' 및 데이터 'b' 를 얻을 수 있다.

[0031] 결국, 제 1 컴퓨터(16)는 단위 시간 동안에 데이터 'a'와 데이터 'b'를 전달받을 수 있다. 마찬가지로, 제 2 컴퓨터(17)는 단위 시간 동안에 데이터 'a'와 데이터 'b'를 전달받을 수 있다. 따라서, 네트워크 코딩 기법을 적용하는 통신망은 높은 데이터 전송 효율을 가질 수 있다.

[0032] 도 2는 본 발명의 기술적 사상의 실시 예에 따른 데이터 패킷들을 보여주는 블록도이다.

[0033] 네트워크 코딩 기법이 적용되는 통신망에 있어서, 송신단은 원본 데이터를 직접 전송하는 대신 원본 데이터를 부호화(encoding)하여 데이터 패킷(data packet)의 형태로 전송한다. 예를 들어 도 2를 참조하면, 데이터 패킷들은 부호화 블록(encoded block) 및 이에 대응하는 부호화 벡터(encoding vector)를 포함한다. 여기서, 부호화 블록(encoded block)은 원본 데이터의 부호화된 조각을 의미한다. 부호화 벡터(encoding vector)는 원본 데이터

를 부호화하기 위한 벡터값을 의미한다.

[0034] 또한, 네트워크 코딩 기법이 적용되는 통신망에 있어서, 수신단은 원본 데이터를 복원하기 위하여 제 1 내지 제 n 데이터 패킷을 전달받는다. 이 경우, 제 1 내지 제 n 부호화 벡터(encoding vector)들의 집합은 부호화 행렬(encoding matrix)로 표현될 수 있다. 제 1 내지 제 n 부호화 블록(encoded block)들의 집합은 부호화 데이터 행렬(encoded data matrix)로 표현될 수 있다.

[0035] 한편, 송신단에서의 부호화(encoding) 동작과 수신단에서의 디코딩(decoding) 동작은 각각 행렬을 이용한 수학식의 형태로 표현될 수 있다. 먼저, 송신단에서의 원본 데이터의 부호화(encoding) 동작은 수학식 1과 같이 표현될 수 있다.

수학식 1

$$\begin{bmatrix} y_1 \\ \vdots \\ y_h \end{bmatrix} = \begin{bmatrix} g_{1,1} & \cdots & g_{1,h} \\ \vdots & \ddots & \vdots \\ g_{h,1} & \cdots & g_{h,h} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix} = G_h \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix}$$

[0036]

[0037] 여기서, yh는 부호화 블록(encoded block)을 나타내고, gh,h는 부호화 벡터(encoding vector)의 원소를 나타내며, xh는 원본 데이터 조각을 나타내며, Gh는 부호화 행렬(encoding matrix)을 나타낸다. 또한, 수학식 1은 수학식 2와 같이 보다 확장된 형태의 행렬식으로 표현될 수 있다.

수학식 2

$$\begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N} \\ \vdots & \vdots & & \vdots \\ y_{h,1} & y_{h,2} & \cdots & y_{h,N} \end{bmatrix} = G_h \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ \vdots & \vdots & & \vdots \\ x_{h,1} & x_{h,2} & \cdots & x_{h,N} \end{bmatrix}$$

[0038]

[0039] 한편, 수신단에서의 디코딩(decoding) 동작은 수학식 3 및 4로 표현될 수 있다.

수학식 3

$$G_h | Y_h = \begin{bmatrix} g_{1,1} & \cdots & g_{1,h} & y_{1,1} & y_{1,2} & \cdots & y_{1,N} \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ g_{h,1} & \cdots & g_{h,h} & y_{h,1} & y_{h,2} & \cdots & y_{h,N} \end{bmatrix}$$

[0040]

수학식 4

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ \vdots & \vdots & & \vdots \\ x_{h,1} & x_{h,2} & \cdots & x_{h,N} \end{bmatrix} = G_h^{-1} \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N} \\ \vdots & \vdots & & \vdots \\ y_{h,1} & y_{h,2} & \cdots & y_{h,N} \end{bmatrix}$$

[0041]

[0042] 수신단에서 디코딩(decoding) 동작을 수행하기 위해서는, 수학식 4에 표현된 것과 같이 부호화 행렬(encoding matrix)의 역행렬(Gh^{-1})을 계산한 후, 이를 부호화 데이터 행렬(encoded data matrix)에 곱해야 한다. 네트워크 코딩 기법을 사용하는 통신망에 있어서, 이러한 디코딩(decoding) 동작은 부호화(encoding) 동작에 비하여 속도가 느린 단점이 있다.

[0043] 이러한 문제점을 해결하기 위하여, 이하에서는 본 발명의 기술적 사상의 실시 예에 따른 디코딩 속도를 빠르게 하기 위한 데이터 복원 장치가 상세히 설명될 것이다.

[0044] 도 3은 본 발명의 기술적 사상의 실시 예에 따른 데이터 복원 장치를 보여주는 블록도이다. 도 3을 참조하면, 데이터 복원 장치(100)는 패킷 저장부(110), 부호화 행렬 변환부(120), 데이터 행렬 변환부(130), 연산 할당부(140), 산술 논리 연산부(150) 및 제어 로직(160)을 포함한다.

[0045] 패킷 저장부(110)는 제 1 메모리(111), 제 2 메모리(112) 및 데이터 분배부(113)를 포함한다. 패킷 저장부(110)는 외부로부터 데이터 패킷(data packet)을 전달받아 저장한다.

[0046] 자세히 설명하면, 데이터 분배부(113)는 외부로부터 데이터 패킷(data packet)을 전달받는다. 데이터 분배부(113)는 데이터 패킷의 부호화 벡터(encoding vector)를 제 1 메모리(111)에 전달한다. 데이터 분배부(113)는 데이터 패킷의 부호화 블록(encoded block)을 제 2 메모리(112)에 전달한다.

[0047] 제 1 메모리(111)에는 데이터 분배부(113)로부터 전달받은 부호화 벡터(encoding vector)가 저장된다. 예를 들어 도 2를 참조하면, 원본 데이터의 데이터 패킷(data packet)들이 모두 수신된 경우, 제 1 메모리(111)에는 제 1 내지 제 n 부호화 벡터들이 저장된다. 이 경우, 제 1 메모리(111)에 저장된 제 1 내지 제 n 부호화 벡터들은 부호화 행렬(encoding matrix)의 형태로 저장될 수 있다.

[0048] 제 2 메모리(112)에는 데이터 분배부(113)로부터 전달받은 부호화 블록(encoded block)이 저장된다. 예를 들어 도 2를 참조하면, 원본 데이터의 데이터 패킷(data packet)들이 모두 수신된 경우, 제 2 메모리(112)에는 제 1 내지 제 n 부호화 블록들이 저장된다. 이 경우, 제 2 메모리(112)에는 제 1 내지 제 n 부호화 블록들이 부호화 데이터 행렬(encoded data matrix)의 형태로 저장될 수 있다.

[0049] 한편, 산술 논리 연산부(150)에서 부호화 행렬(encoding matrix)의 역행렬이 계산된 경우, 제 1 메모리(111)는 부호화 행렬 변환부(120)로부터 역 부호화 행렬(inverse encoding matrix)을 전달받는다. 여기서 역 부호화 행렬(inverse encoding matrix)은 부호화 행렬(encoding matrix)의 역 행렬을 의미한다. 이 경우, 제 1 메모리(111)에는 역 부호화 행렬(inverse encoding matrix)이 저장될 수 있다. 이는 이하에서 좀더 상세하게 설명될 것이다.

[0050] 계속해서 도 3을 참조하면, 부호화 행렬 변환부(120)는 데이터 분배부(113)로부터 부호화 행렬(encoding

matrix)를 전달받는다. 다시 말하면, 제 1 메모리(111)에 저장된 부호화 행렬(encoding matrix)은 데이터 분배부(113)를 통하여 부호화 행렬 변환부(120)에 전달된다. 부호화 행렬 변환부(120)는 전달받은 부호화 행렬(120)을 행(row) 단위로 연산 할당부(140)에 전달한다.

[0051] 한편, 산술 논리 연산부(150)에서 역 부호화 행렬(inverse encoding matrix)이 계산된 경우, 부호화 행렬 변환부(120)는 연산 할당부(140)를 통하여 역 부호화 행렬(inverse encoding matrix)을 전달받아 저장한다. 부호화 행렬 변환부(120)에 저장된 역 부호화 행렬(inverse encoding matrix)은 데이터 분배부(113)를 통하여 제 1 메모리(111)에 저장된다. 부호화 행렬 변환부(120)는 이하의 도 5 및 도 6에서 좀더 상세하게 설명될 것이다.

[0052] 데이터 행렬 변환부(130)는 데이터 분배부(113)로부터 역 부호화 행렬(inverse matrix)을 전달받는다. 다시 말하면, 제 1 메모리(111)에 저장된 역 부호화 행렬(inverse encoding matrix)은 데이터 분배부(113)를 통하여 데이터 행렬 변환부(130)에 전달된다. 데이터 행렬 변환부(130)는 전달받은 역 부호화 행렬(inverse encoding matrix)을 행(row) 단위로 연산 할당부(140)에 전달한다.

[0053] 또한, 데이터 행렬 변환부(130)는 데이터 분배부(113)로부터 부호화 데이터 행렬(encoded data matrix)을 전달받는다. 다시 말하면, 제 2 메모리(112)에 저장된 부호화 데이터 행렬(encoded data matrix)은 데이터 분배부(113)를 통하여 데이터 행렬 변환부(130)에 전달된다.

[0054] 이 경우, 부호화 데이터 행렬(encoded data matrix)은 열(column) 단위로 데이터 분배부(113)에 전달될 수 있다. 데이터 행렬 변환부(130)는 전달받은 열(column) 단위의 부호화 데이터 행렬(encoded data matrix)을 연산 할당부(140)에 전달한다. 데이터 행렬 변환부(130)는 이하의 도 7 및 도 8에서 좀더 상세하게 설명될 것이다.

[0055] 연산 할당부(140)는 부호화 행렬 변환부(120)로부터 부호화 행렬(encoding matrix)을 전달받는다. 이 경우, 연산 할당부(140)는 행(row) 단위로 부호화 행렬(encoding matrix)을 전달받을 수 있다. 연산 할당부(140)는 행 단위(row)의 부호화 행렬(encoding matrix)을 각 원소별로 산술 논리 연산부(150)에 할당한다.

[0056] 연산 할당부(140)는 산술 논리 연산부(150)에서 역 부호화 행렬(inverse encoding matrix)의 계산이 수행된 경우에, 계산된 역 부호화 행렬(inverse encoding matrix)을 부호화 행렬 변환부(120)에 전달한다.

[0057] 또한, 연산 할당부(140)는 데이터 행렬 변환부(130)로부터 행(row) 단위의 역 부호화 행렬(inverse encoding matrix)을 전달받는다. 연산 할당부(140)는 데이터 행렬 변환부(130)로부터 열(column) 단위의 부호화 데이터 행렬(encoded data matrix)을 전달받는다. 연산 할당부(140)는 전달받은 역 부호화 행렬(inverse encoding matrix)과 부호화 데이터 행렬(encoded data matrix)을 산술 논리 연산부(150)에 할당한다.

[0058] 산술 논리 연산부(150)는 제 1 내지 제 n 산술 논리 연산기들(151~15n)을 포함한다. 본 발명의 기술적 사상에 따른 실시 예에 있어서, 산술 논리 연산부(150)는 부호화 행렬(encoding matrix)의 행들(row)의 수와 같은 개수의 산술 논리 연산기들(Arithmetic Logic Unit, ALU)을 포함한다. 따라서, 산술 논리 연산부(150)는 병렬적으로 행(row) 단위의 연산을 수행할 수 있다. 따라서, 산술 논리 연산부(150)는 하나의 산술 논리 연산기(ALU)가 사용되는 경우에 비하여 빠른 속도로 디코딩(decoding) 동작을 수행할 수 있다.

[0059] 자세히 설명하면, 산술 논리 연산부(150)는 가우스-조던 소거법(Gaussian Jordan elimination)에 의하여 역 부호화 행렬(inverse encoding matrix)을 계산할 수 있다. 간략한 설명을 위하여, 부호화 행렬(encoding matrix)은 n*n의 정방 행렬 형태를 갖는다고 가정된다. 이 경우, 산술 논리 연산부(150)는 부호화 행렬(encoding matrix)의 행(row) 또는 열(column)의 개수와 같은 개수의 제 1 내지 제 n 산술 논리 연산기들(151~15n)을 포함할 것이다.

[0060] 가우스-조던 소거법을 수행하는 경우, 제 1 내지 제 n 산술 논리 연산기들(151~15n)은 가우스-조던 소거법을 수행하는 타겟 행(target row)과 피봇 행(pivot row)의 원소 들을 각각 전달받는다.

[0061] 자세히 설명하면, 예를 들어, 타겟 행(target row)이 [a11, a12, .. ,a1n]이고, 피봇 행(pivot row)이 [a21, a22, ..., a2n]이라고 가정된다. 이 경우, 제 1 산술 논리 연산기(151)에는 'a11', 'a21'의 원소가 할당된다. 제 2 산술 논리 연산기(152)에는 'a12', 'a22'의 원소가 할당된다. 마찬가지로, 제 n 산술 논리 연산기(15n)에는 'a1n', 'a2n'의 원소가 할당된다.

[0062] 이 경우, 제 1 내지 제 n 산술 논리 연산기들(151~15n)은 각각 병렬적으로 가우스-조던 소거법을 수행한다. 따라서, 본 발명의 기술적 사상의 실시 예에 따른 산술 논리 연산부(150)는 하나의 산술 논리 연산기(ALU)에 의하여 가우스-조던 소거법이 수행되는 경우에 비하여 빠른 속도로 수행될 수 있다.

[0063] 이와 유사하게, 역 부호화 행렬(inverse encoding matrix)에 부호화 데이터 행렬(encoded data matrix)을 곱하여 원본 데이터를 복원하는 동작이 수행되는 경우, 제 1 내지 제 n 산술 논리 연산기들(151~15n)은 각각 병렬적으로 곱셈 동작을 수행할 수 있다. 따라서, 본 발명의 기술적 사상의 실시 예에 따른 산술 논리 연산부(150)는 하나의 산술 논리 연산기(ALU)에 의하여 곱셈 동작이 수행되는 경우에 비하여 빠른 속도로 수행될 수 있다.

[0064] 한편, 제어 로직(160)은 데이터 복원 장치(100)의 일련의 동작을 제어한다. 자세히 설명하면, 제어 로직(160)은 패킷 저장부(110), 부호화 행렬 변환부(120), 데이터 행렬 변환부(130), 연산 할당부(140) 및 산술 논리 연산부(150)에 제어 신호를 전달하여, 원본 데이터를 복원하는 일련의 동작을 제어한다.

[0065] 도 4는 도 2의 데이터 복원 장치(100)의 동작을 설명하기 위한 순서도이다.

[0066] S110 단계에서, 데이터 패킷(data packet)이 패킷 저장부(110)에 저장된다.

[0067] 자세히 설명하면, 데이터 패킷(data packet)의 부호화 벡터(encoding vector)는 제 1 메모리(111)에 저장된다. 데이터 패킷(data packet)의 부호화 블록(encoded block)은 제 2 메모리(112)에 저장된다. 원본 데이터의 데이터 패킷(data packet)들이 모두 전달된 경우, 제 1 메모리(111)에는 부호화 행렬(encoding matrix)이 저장되고, 제 2 메모리(112)에는 부호화 데이터 행렬(encoded data matrix)이 저장된다.

[0068] S120 단계에서, 부호화 행렬(encoding matrix)의 역 행렬이 생성된다. 즉, 역 부호화 행렬(inverse encoding matrix)이 생성된다.

[0069] 자세히 설명하면, 제 1 메모리(111)에 저장된 부호화 행렬(encoding matrix)이 부호화 행렬 변환부(120)에 전달된다. 부호화 행렬(encoding matrix)은 연산 할당부(140)를 통하여 산술 논리 연산부(150)에 전달된다. 산술 논리 연산부(150)는 역 부호화 행렬(inverse encoding matrix)을 계산한다.

[0070] 이 경우, 제 1 내지 제 n 산술 논리 연산기들(151~15n)은 각각 병렬적으로 동작할 것이다. 따라서, 산술 논리 연산부(150)는 하나의 산술 논리 연산기(ALU)를 사용하는 경우보다 빠른 디코딩(decoding) 동작을 수행할 수 있다. 계산된 역 부호화 행렬(inverse encoding matrix)은 부호화 행렬 변환부(120)에 전달된다.

[0071] S130 단계에서, 역 부호화 행렬(inverse encoding matrix)이 패킷 저장부(110)에 저장된다.

[0072] 자세히 설명하면, 부호화 행렬 변환부(120)에 저장된 역 부호화 행렬(inverse encoding matrix)이 제 1 메모리(111)에 저장된다. 이 경우, 역 부호화 행렬(inverse encoding matrix)은 부호화 행렬(encoding matrix) 위에 덮어 쓰기(over write) 될 수 있다.

[0073] S140 단계에서, 역 부호화 행렬(inverse encoding matrix) 및 부호화 데이터 행렬(encoded data matrix)이 데이터 행렬 변환부(130)에 저장된다.

[0074] 자세히 설명하면, 제 1 메모리(111)에 저장된 역 부호화 행렬(inverse encoding matrix)이 데이터 행렬 변환부(130)에 전달된다. 제 2 메모리(112)에 저장된 부호화 데이터 행렬(encoded data matrix)이 데이터 행렬 변환부(130)에 저장된다. 이 경우, 부호화 데이터 행렬(encoded data matrix)은 열(cloumn) 단위로 데이터 행렬 변환부(130)에 저장될 수 있다.

[0075] S150 단계에서, 원본 데이터가 복원된다.

[0076] 자세히 설명하면, 데이터 행렬 변환부(130)에 저장된 역 부호화 행렬(inverse encoding matrix)이 행(row) 단위로 연산 할당부(140)를 통하여 산술 논리 연산부(150)에 할당된다. 또한, 데이터 행렬 변환부(130)에 저장된 열(column) 단위의 부호화 데이터 행렬(encoded data matrix)이 연산 할당부(140)를 통하여 산술 논리 연산부(150)에 할당된다.

[0077] 산술 논리 연산부(150)는 전달받은 행(row) 단위의 역 부호화 행렬(inverse encoding matrix)과 열(column) 단위의 부호화 데이터 행렬(encoded data matrix)을 곱하여 원본 데이터를 복원한다. 이 경우, 제 1 내지 제 n 산술 논리 연산기들(151~15n)은 각각 병렬적으로 동작할 것이다. 따라서, 산술 논리 연산부(150)는 하나의 산술 논리 연산기(ALU)를 사용하는 경우보다 빠른 디코딩(decoding) 동작을 수행할 수 있다.

[0078] 도 5는 본 발명의 기술적 사상의 실시 예에 따른 도 3의 부호화 행렬 변환부(120)를 좀더 상세하게 보여주는 블록도이다. 도 5를 참조하면, 부호화 행렬 변환부(120)는 제 1 내지 제 4 레지스터들(121~124)을 포함한다.

[0079] 제 1 레지스터(121)는 데이터 분배부(113)로부터 부호화 행렬(encoding matrix)를 전달받는다. 이 경우, 제 1 레지스터(121)의 크기는 부호화 행렬(encoding matrix)의 크기와 동일할 수 있다. 예를 들어, 부호화 행렬

(encoding matrix)이 n*n의 정방 행렬인 경우에, 제 1 레지스터(121)는 n*n의 부호화 행렬(encoding matrix)을 저장할 것이다.

[0080] 제 1 레지스터(121)는 전달받은 부호화 행렬(encoding matrix)의 타겟 행(target row)을 제 3 레지스터(123)에 전달한다. 제 1 레지스터(121)는 전달받은 부호화 행렬(encoding matrix)의 피봇 행(pivot row)을 제 4 레지스터(124)에 전달한다. 여기서, 타겟 행(target row)과 피봇 행(pivot row)은 가우스-조던 소거법(Gaussian Jordan elimination)을 수행하는 경우의 단위가 되는 행(row) 들을 의미한다.

[0081] 제 2 레지스터(122)는 부호화 행렬(encoding matrix)에 대응하는 단위 행렬(identity matrix)을 저장한다. 이 경우, 제 2 레지스터(122)의 크기는 부호화 행렬(encoding matrix)의 크기와 동일할 수 있다. 예를 들어, 부호화 행렬(encoding matrix)이 n*n의 정방 행렬인 경우에, 제 2 레지스터(122)는 n*n의 단위 행렬(identity matrix)을 저장할 것이다.

[0082] 제 2 레지스터(122)는 단위 행렬(identity matrix)의 타겟 행(target row)을 제 3 레지스터(123)에 전달한다. 제 2 레지스터(122)는 단위 행렬(identity matrix)의 피봇 행(pivot row)을 제 4 레지스터(123)에 전달한다.

[0083] 한편, 제 2 레지스터(122)가 단위 행렬(identity matrix)를 저장하므로, 제 2 레지스터(122)에는 가우스-조던 소거법(Gauss Jordan elimination)이 완료된 경우에, 산술 논리 연산부(150)에서 계산된 역 부호화 행렬(inverse encoding matrix)이 저장될 것이다.

[0084] 제 3 레지스터(123)는 제 1 레지스터(121)로부터 부호화 행렬(encoding matrix)의 타겟 행(target row)을 전달받는다. 제 3 레지스터(123)는 제 2 레지스터(122)로부터 단위 행렬(identity matrix)의 타겟 행(target row)을 전달받는다.

[0085] 제 3 레지스터(123)의 크기는 타겟 행(target row)의 크기와 동일 할 수 있다. 이 경우, 제 3 레지스터(123)는 부호화 행렬(encoding matrix)의 타겟 행(target row)과 단위 행렬(identity matrix)의 타겟 행(target row)을 순차적으로 인가받을 수 있다.

[0086] 예를 들어, 부호화 행렬(encoding matrix)의 타겟 행(target row)과 부호화 행렬(encoding matrix)의 피봇 행(pivot row)에 대한 가우스-조던 소거법(Gaussian Jordan elimination)이 수행된 후, 단위 행렬(identity matrix)의 타겟 행(target row)과 단위 행렬(identity matrix)의 피봇 행(pivot row)에 대한 가우스-조던 소거법(Gaussian Jordan elimination)이 수행된다고 가정된다.

[0087] 이 경우에, 먼저, 제 3 레지스터(123)는 부호화 행렬(encoding matrix)의 타겟 행(target row)을 제 1 레지스터(121)로부터 전달받을 것이다. 이 후, 제 3 레지스터(123)는 단위 행렬(identity matrix)의 타겟 행(target row)을 제 2 레지스터(122)로부터 전달받을 것이다.

[0088] 제 4 레지스터(124)는 제 1 레지스터(121)로부터 부호화 행렬(encoding matrix)의 피봇 행(pivot row)을 전달받는다. 제 4 레지스터(124)는 제 2 레지스터(122)로부터 단위 행렬(identity matrix)의 피봇 행(pivot row)을 전달받는다.

[0089] 이 경우, 제 4 레지스터(124)의 크기는 피봇 행(pivot row)의 크기와 동일 할 수 있다. 다시 말하면, 제 4 레지스터(124)는 부호화 행렬(encoding matrix)의 피봇 행(pivot row)과 단위 행렬(identity matrix)의 피봇 행(pivot row)을 순차적으로 인가받을 수 있다. 이는 상술한 제 3 레지스터(123)와 유사하므로, 자세한 설명은 생략될 것이다.

[0090] 한편, 제 3 및 제 4 레지스터(23, 124)에 저장된 데이터는 연산 할당부(140)를 통하여 산술 논리 연산부(150, 도 3 참조)에 전달된다. 이 경우, 연산 할당부(140)는 타겟 행(target row)과 피봇 행(pivot row)의 원소들을 각각 제 1 내지 제 n 산술 논리 연산기(151~15n)에 할당할 것이다.

[0091] 산술 논리 연산부(150)에 의하여 가우스-조던 소거법(Gaussian Jordan elimination)이 수행된 경우, 계산 결과는 제 3 레지스터(123) 및 제 4 레지스터(124)를 통하여 각각 제 1 레지스터(121)에 저장된 부호화 행렬(encoding matrix) 및 제 2 레지스터(122)에 저장된 단위 행렬(identity matrix)에 반영될 것이다.

[0092] 따라서, 가우스-조던 소거법(Gaussian Jordan elimination)의 수행이 완료된 경우, 제 1 레지스터(121)에는 단위 행렬(identity)의 형태의 행렬이 저장될 것이다. 또한, 제 2 레지스터(122)에는 역 부호화 행렬(inverse encoding matrix)의 형태의 행렬이 저장될 것이다.

[0093] 한편, 제 1 레지스터(121)에 저장된 단위 행렬이 표준화된(normalized) 형태가 아닌 경우, 제 2 레지스터(121,

122)에 저장된 역 부호화 행렬(inverse encoding matrix)은 표준화될 것이다.

[0094] 도 6은 도 5의 부호화 행렬 변환부(120)의 동작을 좀더 상세하게 설명하기 위한 순서도이다. 간략한 설명을 위하여, 도 6에서는 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan elimination)이 수행된 후에, 단위 행렬(identity matrix)에 대한 가우스-조던 소거법(Gaussian Jordan elimination)이 수행된다고 가정된다.

[0095] S210 단계에서, 제 1 레지스터(121)에 부호화 행렬(encoding matrix)이 저장된다. 즉, 제 1 메모리(111, 도 3 참조)에 저장된 부호화 행렬(encoding matrix)이 데이터 분배부(113, 도 3 참조)를 통하여 제 1 레지스터(121)에 전달된다.

[0096] S220 단계에서, 제 2 레지스터(122)에 단위 행렬(identity matrix)이 저장된다. 즉, 제 1 레지스터(121)에 저장된 부호화 행렬(encoding matrix)에 대응하는 단위 행렬(identity)이 제 2 레지스터(122)에 저장된다. 이 경우, 단위 행렬(identity matrix)는 제어 로직(160, 도 3 참조)의 제어에 의하여 생성될 수 있다.

[0097] S230 단계에서, 타겟 행(target row)과 피봇 행(pivot row)이 결정된다. 예를 들어, 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 먼저 수행되는 경우, 제 1 레지스터(121)에 저장된 부호화 행렬(encoding matrix)에서 타겟 행(target row) 및 피봇 행(pivot row)이 선택될 것이다.

[0098] S240 단계에서, 제 3 레지스터(123)에 선택된 타겟 행(target row)이 저장될 것이다. 예를 들어, 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 먼저 수행되는 경우, 제 3 레지스터(123)는 제 1 레지스터(121)로부터 선택된 타겟 행(target row)을 전달받을 것이다.

[0099] S250 단계에서, 제 4 레지스터(124)에 선택된 피봇 행(pivot row)이 저장될 것이다. 예를 들어, 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 먼저 수행되는 경우, 제 4 레지스터(124)는 제 1 레지스터(121)로부터 선택된 피봇 행(pivot row)을 전달받을 것이다.

[0100] S260 단계에서, 가우스-조던 소거법(Gaussian Jordan elimination)이 수행될 것이다.

[0101] 구체적으로, 제 3 레지스터(123)에 저장된 선택된 타겟 행(target row)의 원소들과 제 4 레지스터(124)에 저장된 선택된 피봇 행(pivot row)의 원소들이 연산 할당부(140)에 의하여 산술 논리 연산부(150, 도 3 참조)에 할당될 것이다(S261).

[0102] 이 후, 산술 논리 연산부(150)에서 가우스-조던 소거법(Gaussian Jordan elimination)을 수행할 것이다(S262). 이 경우, 제 1 내지 제 n 산술 논리 연산기들(151~15n)은 각각 병렬적으로 연산을 수행할 것이다.

[0103] 산술 논리 연산부(150)에서 가우스-조던 소거법(Gaussian Jordan elimination)이 수행된 후, 연산 결과는 제 1 또는 제 2 레지스터(121, 122)에 반영될 것이다(S263). 예를 들어, 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 먼저 수행되는 경우, 연산 결과는 제 1 레지스터(121)에 반영될 것이다.

[0104] S270 단계에서, 단위 행렬(identity matrix)에 대한 연산이 수행되었는 지가 판별된다. 즉, 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 수행된 후에, 대응하는 단위 행렬(identity matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 수행되었는 지가 판별된다.

[0105] 단위 행렬(identity matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 수행되지 않은 경우, 선택된 타겟 행(target row) 및 피봇 행(pivot row)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 수행될 것이다. 이는 부호화 행렬(encoding matrix)에 대한 연산과 유사하므로, 자세한 설명은 생략될 것이다.

[0106] S280 단계에서, 제 1 레지스터(121)의 부호화 행렬(encoding matrix)이 단위 행렬(identity matrix)로 변환되었는지 판별된다. 다시 말하면, 제 1 레지스터(121)에 저장된 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)의 수행이 완료되었는 지 판별된다.

[0107] 제 1 레지스터(121)의 부호화 행렬(encoding matrix)이 단위 행렬(identity matrix)로 변환된 경우, 제 1 레지스터(121)에는 단위 행렬(identity matrix)이 저장되고, 제 2 레지스터(122)에는 역 부호화 행렬(inverse encoding matrix)이 저장될 것이다. 이 경우, 제 2 레지스터(122)에 저장된 역 부호화 행렬(inverse encoding matrix)은 데이터 분배부(113, 도 3 참조)를 통하여 제 1 메모리(111, 도 3 참조)에 덮어 쓰기(over write)될 것이다.

[0108] 한편, 제 1 레지스터(121)의 부호화 행렬(encoding matrix)이 단위 행렬(identity matrix)로 변환되지 않은 경우, 새로운 타겟 행(target row)과 피봇 행(pivot row)이 결정되고, 상술한 동작이 반복될 것이다.

[0109] 한편, 상술한 동작은 예시적인 것으로 이해되어야 할 것이다. 예를 들어, 본 발명의 기술적 사상에 따른 실시 예에 있어서, 단위 행렬(identity matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 먼저 수행된 후에, 대응하는 부호화 행렬(encoding matrix)에 대한 가우스-조던 소거법(Gaussian Jordan matrix)이 수행될 수 있다.

[0110] 도 7은 본 발명의 기술적 사상의 실시 예에 따른 도 3의 데이터 행렬 변환부(130)를 좀더 상세하게 보여주는 블록도이다. 도 7을 참조하면, 데이터 행렬 변환부(130)는 제 5 내지 제 7 레지스터들(131~133)을 포함한다.

[0111] 제 5 레지스터(131)는 데이터 분배부(113)로부터 역 부호화 행렬(inverse encoding matrix)을 저장한다. 즉, 제 5 레지스터(131)는 제 1 메모리(112, 도 3 참조)에 저장된 역 부호화 행렬(inverse encoding matrix)을 데이터 분배부(113)를 통하여 전달받는다. 이 경우, 제 5 레지스터(131)의 크기는 역 부호화 행렬(inverse encoding matrix)의 크기와 동일할 수 있다.

[0112] 제 6 레지스터(132)는 데이터 분배부(132)로부터 열(column) 단위의 부호화 데이터 행렬(encoded data matrix)을 전달받는다. 즉, 제 6 레지스터(132)는 제 2 메모리(112, 도 3 참조)에 저장된 부호화 데이터 행렬(encoded data matrix) 중 선택된 열(column)을 전달받는다. 이 경우, 제 6 레지스터(132)의 크기는 부호화 데이터 행렬(encoded data matrix)의 열(column)의 크기와 동일할 수 있다.

[0113] 제 7 레지스터(133)는 제 5 레지스터(131)로부터 행(row) 단위의 역 부호화 행렬(inverse encoding matrix)을 전달받는다. 즉, 제 7 레지스터(133)는 제 5 레지스터(131)로부터 역 부호화 행렬(inverse encoding matrix)의 행(row)을 전달받는다. 이 경우, 제 7 레지스터(133)의 크기는 역 부호화 행렬(inverse encoding matrix)의 행(row)의 크기와 동일할 수 있다.

[0114] 한편, 제 6 및 제 7 레지스터(132, 133)에 저장된 데이터는 연산 할당부(140)를 통하여 산술 논리 연산부(150, 도 3 참조)에 할당된다. 이 경우, 연산 할당부(140)는 부호화 데이터 행렬(encoded data matrix) 중 선택된 열(column)의 원소들과 역 부호화 행렬(inverse encoding matrix) 중 선택된 행(row)의 원소들을 각각 제 1 내지 제 n 산술 논리 연산기(151~15n)에 할당할 것이다.

[0115] 이 후, 산술 논리 연산기(150)는 전달받은 데이터를 이용하여 곱셈 연산을 수행할 것이다. 산술 논리 연산기(150)는 곱셈 연산의 결과에 대한 bit-wise XOR 연산을 수행하여 원본 데이터를 복원할 것이다.

[0116] 한편, 도 7에서는 제 2 메모리(112)에 저장된 부호화 데이터 행렬(encoded data matrix)을 저장하기 위한 레지스터가 포함되지 않는다. 이는 부호화 데이터 행렬(encoded data matrix)의 크기가 부호화 행렬(encoding matrix) 또는 역 부호화 행렬(inverse encoding matrix)에 비하여 크기 때문이다. 다만, 이는 예시적인 것으로, 본 발명의 기술적 사상의 다른 실시 예에 따른 데이터 행렬 변환부(130)는 부호화 데이터 행렬(encoded data matrix)을 저장하기 위한 레지스터를 포함할 수 있다.

[0117] 도 8은 도 7의 데이터 행렬 변환부(130)의 동작을 좀더 자세하게 설명하기 위한 순서도이다.

[0118] S310 단계에서, 제 5 레지스터(131)에 역 부호화 행렬(inverse encoding matrix)이 저장된다. 즉, 제 1 메모리(111)에 저장된 역 부호화 행렬(inverse encoding matrix)이 데이터 분배부(113)를 통하여 제 5 레지스터(131)에 전달된다.

[0119] S320 단계에서, 역 부호화 행렬(inverse encoding matrix)의 행(row)과 부호화 데이터 행렬(encoded data matrix)의 열(column)이 선택된다.

[0120] S330 단계에서, 제 6 레지스터(132)에 선택된 부호화 데이터 행렬(encoded data matrix)의 열(column)이 저장된다. 즉, 제 2 메모리(112)에 저장된 부호화 데이터 행렬(encoded data matrix)에서 선택된 열(column)이 데이터 분배부(113)를 통하여 제 6 레지스터(132)에 전달된다.

[0121] S340 단계에서, 제 7 레지스터(133)에 역 부호화 행렬(inverse encoding matrix)의 행(row)이 저장된다. 즉, 제 5 레지스터(131)에 저장된 역 부호화 행렬(inverse encoding matrix)에서 선택된 행(row)이 제 7 레지스터(133)에 전달된다.

[0122] S350 단계에서, 원본 데이터가 복원된다. 즉, 연산 할당부(140)는 제 6 및 제 7 레지스터(132, 133)에 저장된 데이터를 산술 논리 연산부(150, 도 3 참조)에 할당한다. 산술 논리 연산부(150)는 전달받은 데이터에 대한 곱

셈 연산을 각각 병렬적으로 수행한다. 산술 논리 연산부(150)는 곱셈 연산 결과에 대한 bit-wise XOR 연산을 수행하여 원본 데이터를 복원한다.

[0123] S360 단계에서, 모든 데이터가 복원되었는 지의 여부가 판별된다.

[0124] 모든 데이터가 복원되지 않은 경우, 역 부호화 행렬(inverse encoding matrix)의 새로운 행(row)과 부호화 데이터 행렬(encoded data matrix)의 새로운 열(column)이 선택되고, 상술한 과정이 반복된다.

[0125] 한편, 모든 데이터가 복원된 경우, 산술 논리 연산부(150)는 복원된 데이터를 출력한다.

[0126] 도 9는 본 발명의 기술적 사상의 실시 예에 따른 도 3의 산술 논리 연산부(150)의 구조를 보여주는 블록도이다. 도 9에서는 산술 논리 연산부(150)의 예로서, 하나의 산술 논리 연산기(200)가 도시되어 있다.

[0127] 도 9를 참조하면, 산술 논리 연산기(200)는 XOR 게이트(210), AND 게이트(220), 제 1 저장부(230), 제 1 및 제 2 멀티플랙서(240, 250), 제 2 저장부(260), 제 3 저장부(270), 오버플로우 테스터(280) 및 덧셈/뺄셈기(290)를 포함한다.

[0128] XOR 게이트(210)는 제 1 입력 데이터(IN1) 및 제 2 입력 데이터(IN2)를 전달받는다. 여기서, 제 1 및 제 2 입력 데이터(IN1, IN2)는 연산 할당부(140, 도 3 참조)에 의하여 할당되는 각각의 원소를 의미한다. 예를 들어, 산술 논리 연산기(200)에서 가우스-조던 소거법(Gaussian Jordan elimination)을 수행하는 경우, 제 1 입력 데이터(IN1)는 제 3 레지스터(123, 도 3 참조)에 저장된 행렬의 원소들 중 연산 할당부(140)에 의하여 산술 논리 연산기(200)에 할당된 원소를 의미한다. 마찬가지로, 제 2 입력 데이터(IN2)는 제 4 레지스터(124, 도 3 참조)에 저장된 행렬의 원소들 중 연산 할당부(140)에 의하여 산술 논리 연산기(200)에 할당된 원소를 의미한다.

[0129] 이 경우, XOR 게이트(210)는 덧셈 연산 또는 뺄셈 연산을 수행한다. 이는 본 발명의 기술적 사상의 실시 예에 따른 네트워크 코딩에서의 각 원소들은 유한 필드(finite field or galois field) 내의 숫자로 취급되며, 유한 필드에서 덧셈 및 뺄셈은 bit-wise XOR 연산으로 수행되기 때문이다.

[0130] AND 게이트(220)는 제 1 입력 데이터(IN1) 및 제 2 입력 데이터(IN2)를 전달받는다. AND 게이트(220)는 제 1 및 제 2 입력 데이터(IN1, IN2) 중 어느 하나가 '0'의 데이터를 갖는 경우에 '0'의 출력 데이터를 갖는다. 따라서 AND 게이트(220)는 제 1 입력 데이터(IN1) 또는 제 2 입력 데이터(IN2) 중 어느 하나가 '0'의 데이터를 갖는지 검사할 수 있다.

[0131] 곱셈 또는 나눗셈 연산을 로그(log) 치환을 통하여 수행할 때, 입력 데이터가 '0'일 경우 잘못된 결과가 출력될 수 있다. 따라서, 이 경우 AND 게이트(220)는 계산 결과를 강제로 '0'으로 만들어 준다.

[0132] 제 2 및 제 3 저장부(260, 270)는 각각 제 1 및 제 2 입력 데이터(IN1, IN2)를 전달받는다. 제 2 및 제 3 저장부(260, 270)는 각각 로그 테이블(log table)을 저장한다. 따라서, 곱셈 또는 나눗셈 연산이 수행되는 경우, 제 2 및 제 3 저장부(260, 270)는 제 1 및 제 2 입력 데이터들(IN1, IN2)을 각각 로그(log) 값으로 변환한다. 이 경우, 곱셈 또는 나눗셈 연산은 각각 덧셈 및 뺄셈 연산으로 수행될 수 있다.

[0133] 오버플로우 테스터(280)는 제 2 및 제 3 저장부들(260, 270)로부터 제 1 및 제 2 로그 입력 데이터들(LIN1, LIN2)을 전달받는다. 여기서 제 1 및 제 2 로그 입력 데이터들(IN1, IN2)은 각각 로그 치환된 제 1 및 제 2 입력 데이터들(IN1, IN2)을 의미한다.

[0134] 오버플로우 테스터(280)는 제 1 및 제 2 로그 입력 데이터들(LIN1, LIN2)의 오버플로우(overflow) 유무를 검사한다. 오버플로우(overflow)가 존재하는 경우, 오버플로우 테스터(280)는 오버플로우 신호(OVF)를 덧셈/뺄셈기(290)에 전달한다.

[0135] 덧셈/뺄셈기(290)는 제 2 및 제 3 저장부들(260, 270)로부터 제 1 및 제 2 로그 입력 데이터들(LIN1, LIN2)을 전달받는다. 덧셈/뺄셈기(290)는 인가받은 제 1 및 제 2 로그 입력 데이터들(LIN1, LIN2)을 이용하여 덧셈 또는 뺄셈 연산을 수행한다. 이 경우, 제 1 및 제 2 로그 입력 데이터들(LIN1, LIN2)의 덧셈 또는 뺄셈 연산은 제 1 및 제 2 입력 데이터들(IN1, IN2)의 곱셈 또는 나눗셈 연산에 각각 대응한다.

[0136] 또한, 덧셈/뺄셈기(290)는 오버플로우 신호(OVF)를 인가받은 경우에, 오버플로우(overflow)를 보정하기 위한 연산을 함께 수행할 수 있다.

[0137] 한편, 제 1 저장부(230)는 덧셈/뺄셈기(290)로부터 덧셈 또는 뺄셈 연산의 결과 값을 전달받는다. 제 1 저장부(230)는 로그 테이블(log table)을 저장하고 있다. 제 1 저장부(230)는 인가받은 덧셈 또는 뺄셈 연산의 결과를

역로그(anti log) 치환한다. 역로그(anti log) 치환된 데이터는 제 1 및 제 2 멀티플랙서들(240, 250)을 통과하여 원본 데이터로 복원된다.

[0138] 본 발명의 범위 또는 기술적 사상을 벗어나지 않고 본 발명의 구조가 다양하게 수정되거나 변경될 수 있음은 이 분야에 숙련된 자들에게 자명하다. 상술한 내용을 고려하여 볼 때, 만약 본 발명의 수정 및 변경이 아래의 청구항들 및 동등물의 범주 내에 속한다면, 본 발명이 이 발명의 변경 및 수정을 포함하는 것으로 여겨진다.

부호의 설명

[0139]

10: 통신망 20: 데이터 패킷들

100: 데이터 복원 장치 110: 패킷 저장부

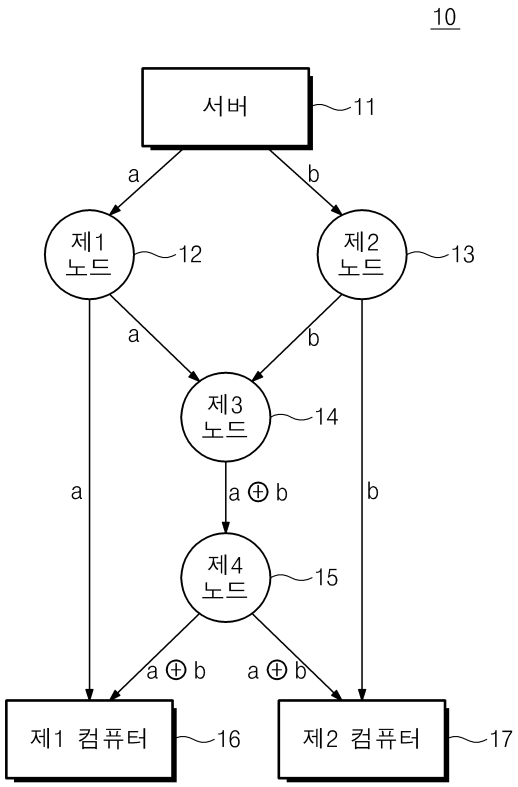
120: 부호화 행렬 변환부 130: 데이터 행렬 변환부

140: 연산 할당부 150: 산술 논리 연산부

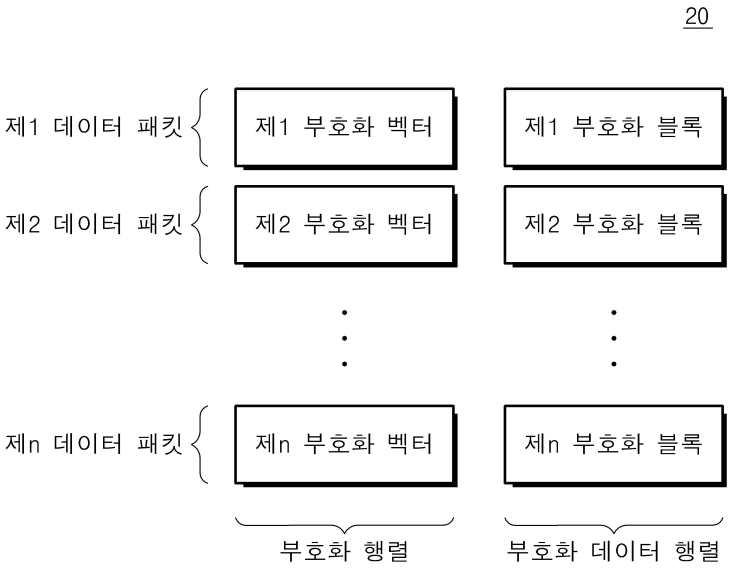
160: 제어 로직

도면

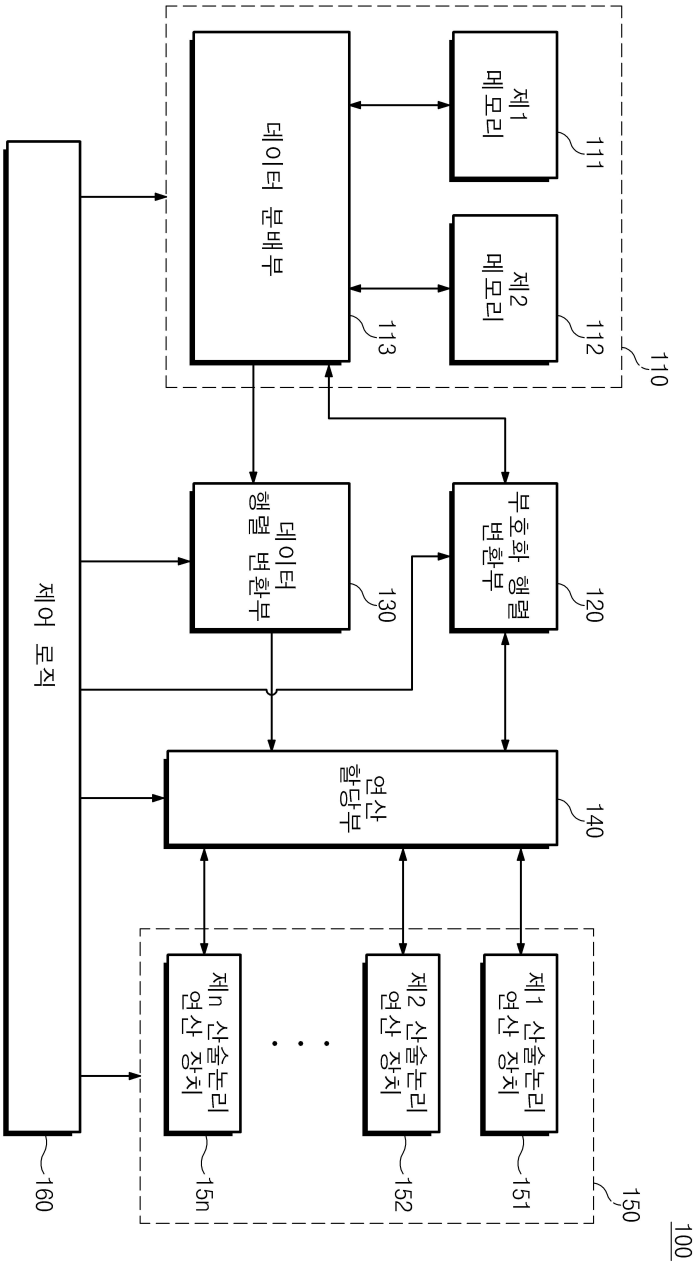
도면1



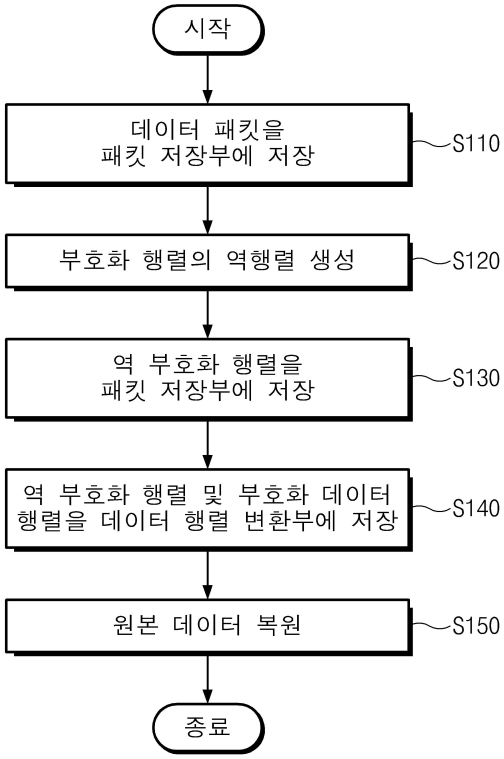
도면2


20
제1 데이터 패킷
제1 부호화 벡터
제1 부호화 블록
제2 데이터 패킷
제2 부호화 벡터
제2 부호화 블록
제n 데이터 패킷
제n 부호화 벡터
제n 부호화 블록
부호화 행렬
부호화 데이터 행렬

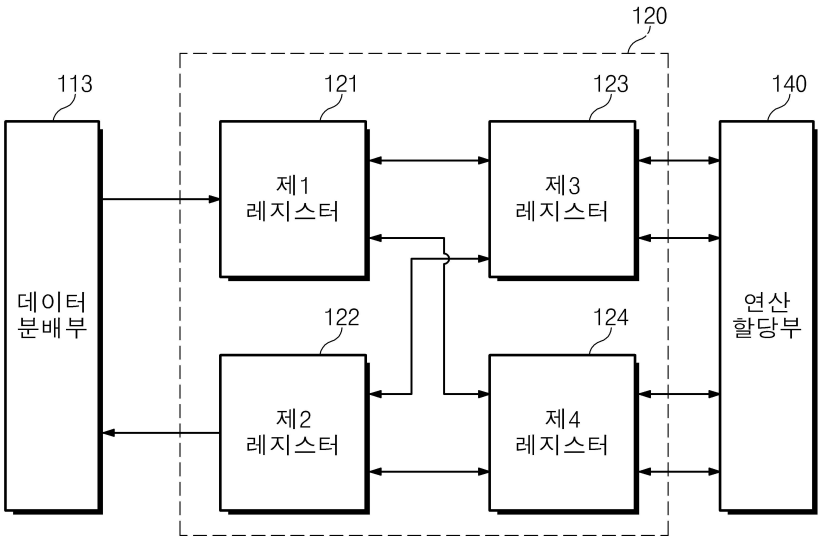
도면3


제어 로직
160
데이터 분배부
113
제1 메모리
111
제2 메모리
112
110
데이터 행렬 변환부
130
부호화 행렬 변환부
120
연산 할당부
140
제n 산술논리 연산 장치
15n
제2 산술논리 연산 장치
152
제1 산술논리 연산 장치
151
150
100

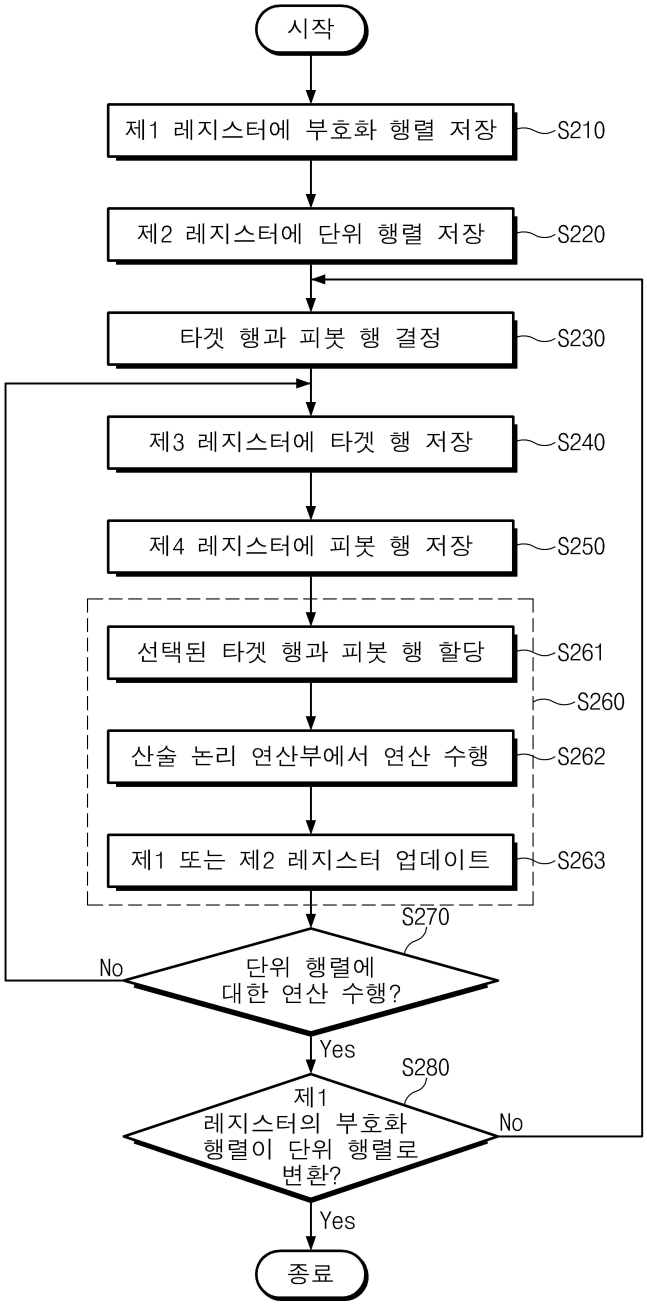
도면4


시작
데이터 패킷을
패킷 저장부에 저장
S110
부호화 행렬의 역행렬 생성
S120
역 부호화 행렬을
패킷 저장부에 저장
S130
역 부호화 행렬 및 부호화 데이터
행렬을 데이터 행렬 변환부에 저장
S140
원본 데이터 복원
S150
종료

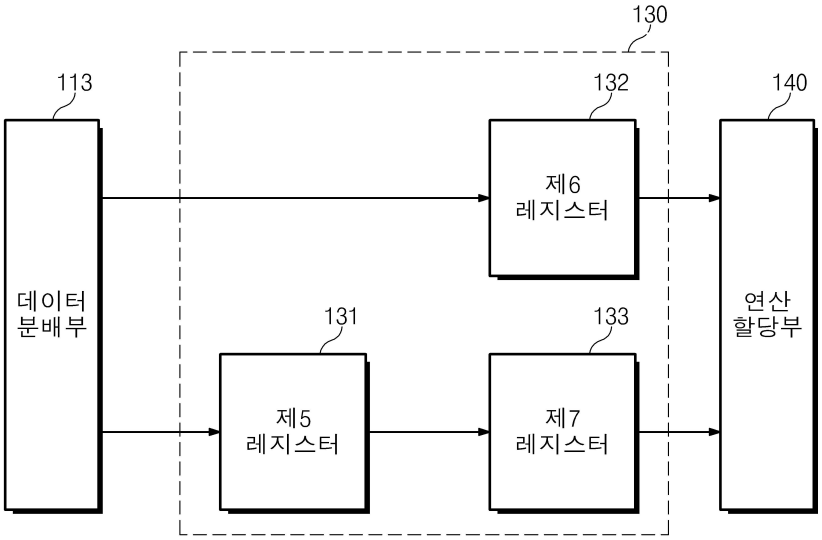
도면5


120
113
121
123
140
데이터
분배부
제1
레지스터
제3
레지스터
122
124
연산
할당부
제2
레지스터
제4
레지스터

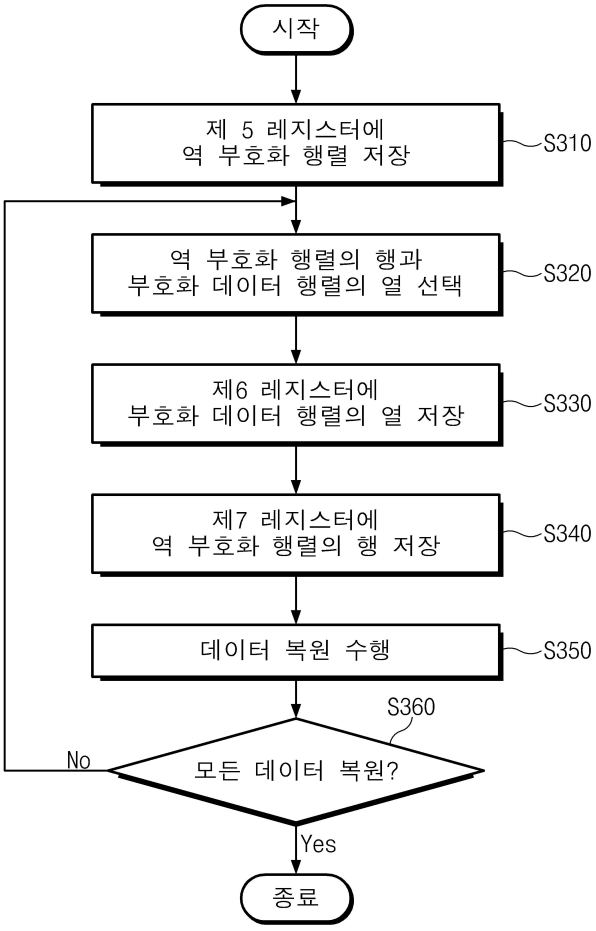
도면6


시작
제1 레지스터에 부호화 행렬 저장
S210
제2 레지스터에 단위 행렬 저장
S220
타겟 행과 피봇 행 결정
S230
제3 레지스터에 타겟 행 저장
S240
제4 레지스터에 피봇 행 저장
S250
선택된 타겟 행과 피봇 행 할당
S261
S260
산술 논리 연산부에서 연산 수행
S262
제1 또는 제2 레지스터 업데이트
S263
S270
단위 행렬에 대한 연산 수행?
No
Yes
S280
제1 레지스터의 부호화 행렬이 단위 행렬로 변환?
No
Yes
종료

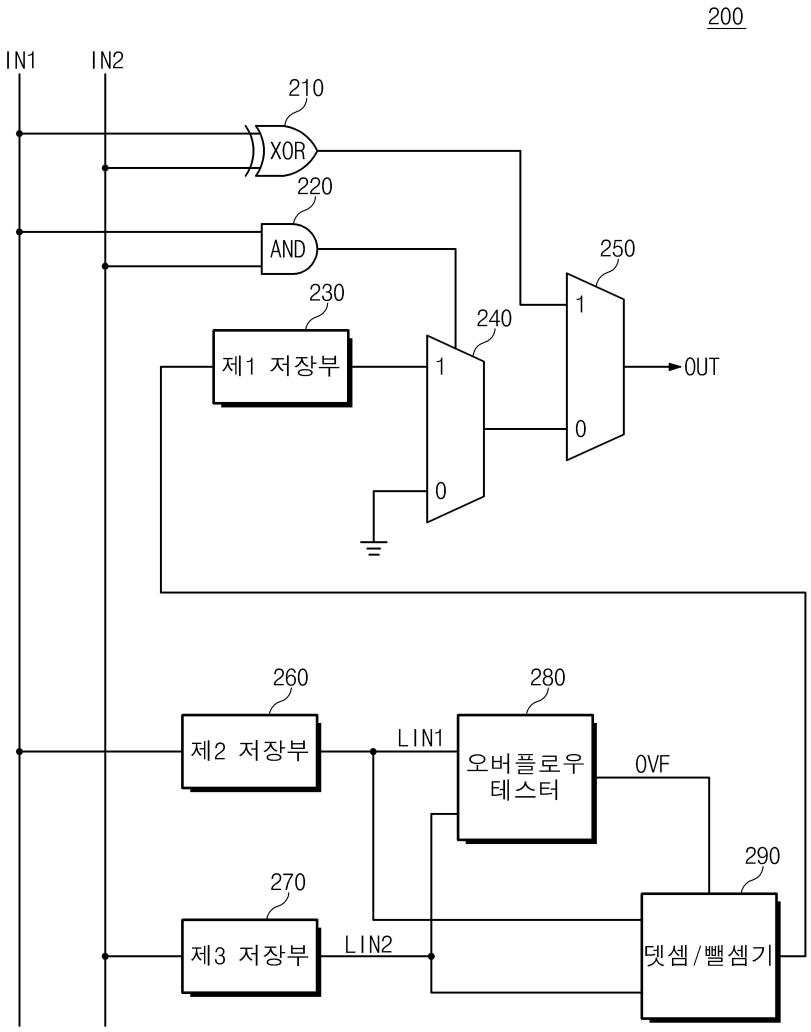
도면7


130
113
132
140
제6
레지스터
데이터
분배부
연산
할당부
131
133
제5
레지스터
제7
레지스터

도면8


시작
제 5 레지스터에
역 부호화 행렬 저장
S310
역 부호화 행렬의 행과
부호화 데이터 행렬의 열 선택
S320
제6 레지스터에
부호화 데이터 행렬의 열 저장
S330
제7 레지스터에
역 부호화 행렬의 행 저장
S340
데이터 복원 수행
S350
S360
No
모든 데이터 복원?
Yes
종료

도면9


200
IN1
IN2
210
XOR
220
AND
230
제1 저장부
240
250
1
0
OUT
260
제2 저장부
LIN1
280
오버플로우
테스터
OVF
290
270
제3 저장부
LIN2
뎃셈/뺄셈기