(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2009년11월09일
(11) 등록번호 10-0926207
(24) 등록일자 2009년11월03일

(51) Int. Cl.
H04N 7/24 (2006.01) *H04N 7/30* (2006.01)
(21) 출원번호 10-2007-0086206
(22) 출원일자 2007년08월27일
심사청구일자 2007년08월27일
(65) 공개번호 10-2008-0020947
(43) 공개일자 2008년03월06일
(30) 우선권주장
11/510,218 2006년08월25일 미국(US)
(56) 선행기술조사문헌
WO2005038719 A2
KR1020030027321 A
KR1020060016947 A
전체 청구항 수 : 총 10 항

(73) 특허권자
엔비디아 코포레이션
미국 캘리포니아 95050 산타 클라라 산 토마스 익스프레스웨이 2701
(72) 발명자
불루수 라비
미국 95054 캘리포니아주 산타클라라 넘버310 맨션 코트 530
스리람 파르타
미국 94022 캘리포니아주 로스 알토스 알몬드 애비뉴 93
(74) 대리인
백만기, 양영준

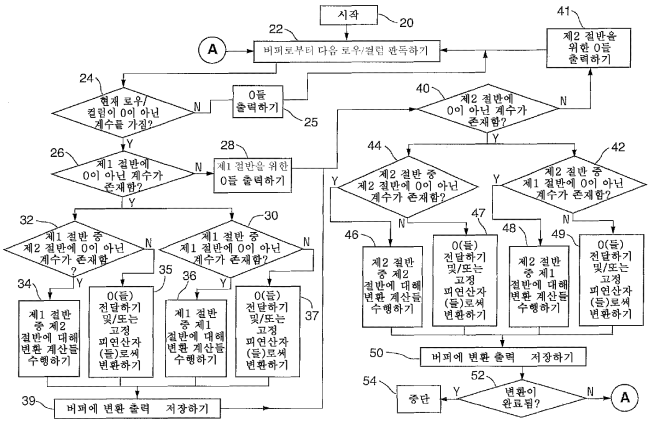
심사관 : 박상철

(54) 전력 소비를 감소시키면서 데이터 값 배열에 대해 2-차원변환을 수행하기 위한 방법 및 시스템

(57) 요 약

일부 실시예들에서, 데이터 블록들의 시퀀스의 각 블록에 대해 2D 변환(예를 들어, IDCT)을 수행하기 위한 방법 및 2D 변환을 수행하도록 구성된 시스템이 제공되는데, 여기에서, 2D 변환은 로우 변환 및 컬럼 변환을 포함한다. 데이터의 로우 또는 컬럼에 대해 로우 또는 컬럼 변환을 수행하기 위해, 이 실시예들은, 로우(컬럼)의 일부분을 구비하는 데이터 값들의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부, 파티션의 제1 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부, 및 파티션의 하나 이상의 다른 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정한다. 하나 이상의 0 값 및 하나 이상의 0이 아닌 값을 포함하는 각각의 로우 또는 컬럼에 대해 로우(컬럼) 변환을 수행할 때, 하나 이상의 0 값에 대한 하나 이상의 변환 동작은 바이패스되거나 감소된 전력 방식으로 수행되는데, 여기에서, 그러한 변환 동작은, 0 값이 0이 아닌 값이라면, 다르게는 전체 전력을 소비하는 방식으로 수행될 것이다. 일부 실시예들에서, 시스템은 발명에 따라 구현된 변환 엔진을 포함하는 파이프라인형 비디오 디코딩 시스템 또는 다른 비디오 프로세싱 시스템(이나 휴대용 미디어 플레이어 또는 다른 시스템의 비디오 프로세싱 서브시스템)이다. 다른 태양들은 시스템의 임의 실시예에서 사용하기 위한 변환 엔진들 및 변환 엔진 회로이다.

대 표 도



특허청구의 범위

청구항 1

입력 데이터 블록들의 시퀀스에서 입력 데이터 블록 각각에 대해 2D 변환을 수행하도록 구성된 시스템으로서, 상기 입력 데이터 블록 각각은 입력 데이터 값들의 로우들 및 컬럼들을 포함하고, 상기 2D 변환은 로우 변환 및 컬럼 변환을 포함하며, 상기 시스템은,

상기 입력 데이터 블록 각각에 응답하여 상기 입력 데이터 블록의 로우 각각에 대해 상기 로우 변환을 수행함으로써 부분 변환된 데이터 블록을 발생시키도록 구성된 로우 변환 회로; 및

상기 부분 변환된 데이터 블록의 컬럼 각각에 대해 상기 컬럼 변환을 수행하도록 구성된 컬럼 변환 회로를 포함하고,

상기 로우 변환 회로는, 로우의 제1 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하도록 구성되며,

상기 로우 변환 회로는, 또한 상기 로우의 제2 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하도록 구성되고,

상기 로우 변환 회로는, 또한 상기 로우 변환을 수행할 때, 0 값을 포함하는 하나 이상의 데이터 값과 연관되어 선택된 동작을 수행하도록 구성되며, 상기 하나 이상의 데이터 값은 상기 로우의 상기 제1 서브세트와 상기 로우의 상기 제2 서브세트로 구성되는 그룹으로부터 선택된 데이터 값을 포함하는

시스템.

청구항 2

제1항에 있어서,

상기 선택된 동작은 하나 이상의 변환 동작을 바이패스하는 것을 포함하는

시스템.

청구항 3

제1항에 있어서,

상기 선택된 동작은 0 값을 포함하는 상기 하나 이상의 데이터 값에 대하여 하나 이상의 변환 동작을 감소된 전력 방식으로 수행하는 것을 포함하는

시스템.

청구항 4

제1항에 있어서,

상기 로우 변환 회로와 상기 컬럼 변환 회로에 결합된 버퍼를 더 포함하고, 상기 로우 변환 회로는 상기 부분 변환된 데이터 블록을 상기 버퍼에 저장하도록 구성되며 상기 컬럼 변환 회로는 상기 부분 변환된 데이터 블록의 컬럼들을 상기 버퍼로부터 불러 오도록 구성되는 시스템.

청구항 5

제1항에 있어서,

상기 로우 변환 회로는, 상기 로우 각각에 대해 상기 로우 변환을 수행할 때, 재순서화 동작을 수행하도록 구성되고, 상기 재순서화 동작은 상기 로우의 데이터 값들을 재순서화하는 것에 의해 재순서화된 데이터 값들을 발생시키며,

상기 로우 변환 회로는, 또한 상기 재순서화된 데이터의 제1 서브세트가 하나 이상의 0 값을 포함하는지의 여부 및 상기 재순서화된 데이터의 제2 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하도록 구성되고, 상기 재순서화된 데이터의 상기 제2 서브세트는 상기 재순서화된 데이터의 상기 제1 서브세트와는 구별되

는 시스템.

청구항 6

입력 데이터 블록들의 시퀀스에 대해 2D 변환을 수행하기 위한 방법으로서, 상기 블록들 각각은 입력 데이터 값들의 로우들 및 컬럼들을 포함하고, 상기 2D 변환은 로우 변환 및 컬럼 변환을 포함하며, 상기 방법은,

부분 변환된 데이터 블록을 발생하도록 상기 입력 데이터 블록들 중 하나의 로우 각각에 대해 상기 로우 변환을 수행하는 단계로서, 상기 로우 변환을 수행하는 상기 단계는 상기 로우 각각의 입력 데이터 값들에 대해 변환 동작들을 수행하는 것을 포함하는 단계; 및

상기 부분 변환된 데이터 블록의 컬럼 각각에 대해 상기 컬럼 변환을 수행하는 단계로서, 상기 컬럼 변환을 수행하는 상기 단계는 상기 컬럼 각각의 데이터 값들에 대해 추가적인 변환 동작들을 수행하는 것을 포함하는 단계

를 포함하고,

상기 로우 변환을 수행하는 상기 단계는,

로우의 제1 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계;

상기 로우의 제2 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계; 및

0 값을 포함하는 하나 이상의 데이터 값과 연관되어 선택된 동작을 수행하는 단계 - 상기 하나 이상의 데이터 값은 상기 로우의 상기 제1 서브세트와 상기 로우의 상기 제2 서브세트로 구성되는 그룹으로부터 선택된 데이터 값을 포함함 -

를 더 포함하는

입력 데이터 블록들의 시퀀스에 대해 2D 변환을 수행하기 위한 방법.

청구항 7

제6항에 있어서,

상기 로우 변환을 수행하는 상기 단계는, 제1 입력 및 제2 입력을 가진 곱셈 회로를, 상기 제1 입력으로 어서트되는 제1 값이 상기 제2 입력으로 어서트되는 제2 값만큼 곱셈되는 곱셈 동작을 수행하도록 동작시키는 단계를 더 포함하고,

선택된 동작을 수행하는 상기 단계는, 0 값을 포함하는 상기 하나 이상의 데이터 값을 위한 상기 곱셈 동작을, 상기 제1 입력 및 상기 제2 입력 중 하나 이상의 토글링을 방지하여, 감소된 전력 방식으로 수행하는 단계를 더 포함하는 방법.

청구항 8

제6항에 있어서,

상기 선택된 동작은, 하나 이상의 변환 동작을 바이패스하는 것과, 0 값을 포함하는 상기 하나 이상의 데이터 값에 대하여 하나 이상의 변환 동작을 감소된 전력 방식으로 수행하는 것으로 구성되는 그룹으로부터 선택되는 방법.

청구항 9

제6항에 있어서,

상기 컬럼 변환을 수행하는 상기 단계는,

컬럼의 제1 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계;

상기 컬럼의 제2 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계; 및

하나 이상의 변환 동작을 바이패스하는 단계를 더 포함하는 방법.

청구항 10

제6항에 있어서,

상기 컬럼 변환을 수행하는 상기 단계는,

컬럼의 제1 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계;

상기 컬럼의 제2 서브세트가 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계; 및

0값을 포함하는 상기 하나 이상의 데이터 값에 대하여 상기 추가적인 변환 동작들 중 하나 이상을 감소된 전력 방식으로 수행하는 단계

를 더 포함하는 방법.

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

청구항 48

삭제

청구항 49

삭제

청구항 50

삭제

청구항 51

삭제

청구항 52

삭제

청구항 53

삭제

청구항 54

삭제

청구항 55

삭제

청구항 56

삭제

청구항 57

삭제

명 세 서

발명의 상세한 설명

기 술 분 야

<1> 본 발명은 데이터 값들의 2-차원(2D) 배열들에 대해 2D 변환들(예를 들어, IDCT들(inverse discrete cosine transforms) 또는 IHT들(inverse Hadamard transforms))을 수행하기 위한 방법들 및 시스템들에 관한 것이다. 통상적인 실시예들에서, 본 발명은, 0 값들(zero values)을 변환하는데 필요치 않은 전력-소비 동작들을 방지하는 것에 의해 전력 소비를 감소시키는 방식으로, 각각이 0 값일 유의 확률(significant probability)을 가진 이미지 데이터 값들의 2D 어레이에 대해 2D 직교 변환의 역을 수행하기 위한 방법들 및 회로에 관한 것이다.

배 경 기 술

<2> 설명서 전체에 걸쳐, 비디오 데이터의 "블록"이라는 용어는 프레임의 직사각형 영역내에 공간적 위치를 가지고 있는 비디오 데이터의 프레임을 구비하는 데이터의 서브세트를 지시하는데 사용된다. 비디오 데이터의 블록은 압축된(또는 인코딩된) 비디오 데이터로 이루어질 수 있지만, 반드시 그래야 하는 것은 아니다. 비디오 데이터의 블록들의 일례들로는, 전통적으로 정의된 MPEG-인코딩형 비디오 프레임들의 매크로블록들을 들 수 있다.

<3> 전통적인 다수 애플리케이션들에서, 이미지 데이터(예를 들어, 비디오 데이터) 또는 다른 데이터는 2D 변환을 경험하고, 변환된 데이터는 나중에 역변환되어 원래 데이터를 복구한다. 그러한 변환들의 일례들로는 2D DCT들(discrete cosine transforms), 2D 하다마드 변환들(Hadamard transforms), 및 2D 푸리에 변환들을 들 수 있다.

<4> 설명서 전체에 걸쳐, (다르게는 동작 출력 값을 발생시킬) 동작을 "바이패스한다"는 표현은, 동작을 실제로 수행하지 않으면서, (동작 출력 값 대신에) 대체 출력 값을 발생시키거나 어서트(aasert)하는 것을 지시한다. 곱셈 회로가 그것의 출력에서 전류(cz1)를 어서트하도록 하기 위해 곱셈 회로의 입력들에 0 값(z1) 및 0이 아닌 상수(c)를 어서트하고, 제2 곱셈 회로가 그것의 출력에서 전류(dz2)를 어서트하도록 하기 위해 제2 곱셈 회로의 입력들에 다른 0 값(z2) 및 0이 아닌 다른 상수(d)를 어서트하며, 전류들(cz1 및 dz2)에 응답하여 노드에서 (그라운드 전위 이상의 0 볼트와 동일한) 출력 전압(cz1+dz2)을 어서트하도록 덧셈 회로를 조작하는 동작을 "바이패스하는" 일례는, 곱셈 회로들 및 덧셈 회로에서 곱셈 및 덧셈 단계들을 실제로 수행하지 않으면서, 노드를 접지시킬 것이다(그것에 의해 노드는 그라운드 전위(ground potential)를 갖게 된다).

<5> 본 발명은, 각각의 값이 0 값이 되는 유의 확률을 가지는 데이터 값들의 2D 배열들(즉, 데이터 값들의 로우들 및 컬럼들로 이루어진 배열들)에 대해 2D 변환들을 수행하기 위한 향상된 방법들 및 시스템들에 관한 것이다. 통상적인 실시예들에서, 본 발명은, 각각의 값이 0 값이 되는 유의 확률을 가지는 데이터 값들의 2D 배열에 대해 2D 직교 변환의 역변환(예를 들어, 2D IDCT 또는 IHT)을 수행하기 위한 향상된 방법 및 시스템에 관한 것이다. 바람직한 실시예들의 부류에서, 본 발명은 DCT 계수들에 대해 2차원 IDCT(2D IDCT)를 수행하기 위한 향상된 방법 및 시스템에 관한 것이다. DCT 계수들은 비디오 데이터(또는 다른 이미지 데이터)의 배열에 대해 2D DCT를 수행하는 것에 의해 발생되었고, 각각이 0 값을 가질 유의 확률을 가진다.

<6> 이 설명서 전체에 걸쳐, "0 값"(또는 "0 데이터 값")이라는 표현은 값 0을 나타내는 데이터를 지시한다. 마찬가지로, "0 입력 데이터 값"은 값 0을 나타내는 입력 데이터를 지시한다. 예를 들어, 0 입력 값은 값 0을 가진 입력 데이터의 워드(예를 들어, DCT 계수, 또는 비디오 데이터의 컬러 성분이나 픽셀)일 수 있다.

<7> 이 설명서 전체에 걸쳐, "스파스(sparse)" 데이터(예를 들어, 역변환을 겪을 데이터의 스파스 블록)는 0 값들일 것 같은 값들을 나타내는 데이터를 지시한다. 예를 들어, 상대적으로 많은 0 값들 및 상대적으로 적은 0이 아닌 값들을 나타내는 입력 데이터의 블록(예를 들어, DCT 계수들의 블록)이 데이터의 스파스 블록이다.

<8> 역변환 구현은 통상적으로, 임의의 비디오 압축 및 압축 해제 표준을 준수할 임의 시스템의 구현의 주된 부분(major part)이다. 그것은 계산 집약적 프로세스이고 프로세싱 사이클 및 전력 소비 요구 사항들에 크게 기여한다. 비디오 압축 및 압축 해제 표준들을 구현하는 모바일 디바이스들(예를 들어, 휴대용 미디어 플레이어들)은 특히 엄격한 프로세싱 사이클 및 전력 소비 요구 사항들을 가지며; 모바일 디바이스들은 애플리케이션에 의해 설정된 엄격한 실행 요구 사항들을 충족시켜야 하고 아주 낮은 전력을 소비하여 배터리 수명을 최대한 늘려야 하며; 변환 엔진은 통상적으로 다수 압축 표준들 및 이 표준들에 수반되는 가변 요구 사항들을 지원할 수 있어야 한다.

<9> 데이터의 블록들에 대한 (2D 역변환들을 포함하는) 2D 변환들의 통상적이고 전통적인 구현들은 다음의 기술들을 상이한 조합들로 사용하여 성능을 향상시키거나 전력을 감소시킨다:

<10> 1. (변환 엔진에 제공되는 각각의 입력 블록이 외부 수단에 의해 코딩 또는 언코딩 블록으로서 식별되는 경우) 외부 수단에 의해 언코딩 블록들로서 식별되는 블록들의 변환 방지하기. 그러나, 이 기술은, 불필요한 변환 동작들(예를 들어, 코딩 블록들로서 식별되지만 0 DC 계수들만으로 이루어진 블록들의 변환)의 수행을 초래할 수 있다는 것을 포함하여, 단점들을 가진다.

<11> 2. 전체적으로 0 값들로 이루어진 각 입력 데이터 블록의 전체 로우들 또는 컬럼들("0-로우들" 또는 "0-컬럼들")을 식별하고, 다르게는 (예를 들어, 각각의 0-로우 또는 0-컬럼을 위해 소정 값들, 통상적으로 "0"을 출력하는 것에 의해) 그러한 로우 또는 컬럼 각각에 대해 수행될 정규 변환 동작들을 바이패스하기. 0-로우들 및 0-컬럼들은 외부 디바이스에 의해 특정되거나 변환 엔진에 의해 내부적으로 식별될 수 있다. 그러나, 전통적인 이 기술은, 로우(또는 컬럼)가 0-로우(또는 0-컬럼)는 아니지만 극소수의 0이 아닌 값들만을 포함하는 스파스하게 채워진 로우(또는 컬럼)인, 흔히 일어나는 다수 상황들에서 성능을 향상시키거나 전력을 감소시키지 않는다.

<12> 3. (입력 데이터 로우 또는 컬럼에서의 데이터 값들을 곱셈하는데 사용하기 위해 앞서 결정된) 동일한 계수들이 후속적인 입력 데이터 로우 또는 컬럼에서의 데이터 값들을 곱셈하는데도 사용되어야 한다는 것을 나타내는 조건들을 (입력 데이터로부터) 식별하고, 다르게는 후속적인 입력 데이터 로우 또는 컬럼에서의 데이터 값들을 곱셈하기 위한 새로운 계수들을 결정하기 위해 수행될 그러한 계수들의 업데이트를 방지하기.

<13> 4. (2D 변환의 룩업 테이블-기반 구현인) 분산형 산술 변환 구현하기. 통상적인 룩업 테이블-기반 구현은, 블록을 변환하기 위해 수행되어야 하는 곱셈 동작들의 수를 감소시키는 것에 의해 오버헤드를 감소시킨다. 그러나, 그러한 구현을 설계하기 위해서는 통상적으로, 아주 큰 ROM 테이블들 및 멀티-포트형(multi-ported) ROM이 통상적으로 요구되기 때문에, 아주 복잡하고, 설계 요구 사항들은 통상적으로, 실현될 수 있는 전력 소비에서의 향상을 제한한다.

<14> 미국 특허출원공개 제2005/0033788호 및 관련된 미국특허 제6,799,192호에서 설명된 전통적인 다른 2D 변환에서는, (IDCT의 컬럼 변환 단계를 수행할 때) 데이터 블록의 각 컬럼에서의 0이 아닌 마지막 엔트리가 판정된 다음, 변환 시스템은 각 컬럼을 역변환하기 위한 IDCT 동작들을 소프트웨어로 구현하기 위한 8개의 상이한 "특수 IDCT(specialized IDCT)" 프로그램 루틴들 중 적합한 하나로 분기한다. 명시적으로, (좀더 적은 곱셈 및 덧셈 동작들을 요구하는) 좀더 간단한 변환 동작들이 (좀더 높은 위치(higher position)에서 0이 아닌 마지막 값을 갖는 것에 의해 지시되는) 상대적으로 많은 0들을 가진 컬럼을 프로세싱하는데 이용될 수 있고, (좀더 많은 곱셈 및 덧셈 동작들을 요구하는) 좀더 복잡한 변환 동작들이 (좀더 낮은 위치에서 0이 아닌 마지막 값을 갖는 것에 의해 지시되는) 좀더 적은 0들을 가진 컬럼을 프로세싱하는데 이용될 수 있다. 참고 문헌들은, (컬럼 변환 단계 이후에) IDCT의 로우 변환 단계를 수행할 때, 블록의 각 로우에서의 0이 아닌 마지막 엔트리가 판정된 다음 변환 시스템은 각 로우를 역변환하기 위한 IDCT 동작들을 소프트웨어로 구현하기 위한 8개의 상이한 "특수 IDCT" 프로그램 루틴들 중 적합한 하나로 분기한다는 것도 교수한다.

<15> 그것이, 0값 및 0이 아닌 값들(zero and non-zero values)의 통상적인 패턴들을 가진 다수 컬럼들 및 로우들을 프로세싱할 때, 성능을 향상시키거나 전력 소비를 감소시키지 않는다는 의미에서 그 기술이 비효율적이라는 것을 포함하여, 미국 특허출원공개 제2005/0033788호 및 미국특허 제6,799,192호에서 설명된 기술에는 여러가지 문제점들 및 제약들이 존재한다. 예를 들어, 변환될 컬럼 또는 로우가 0들(특히, 많은 0들)을 포함하지만 0이 아닌 마지막 엔트리를 가질 경우, 이 기술은 컬럼 또는 로우를 변환하기 위해 많은 전력을 소비하는 복잡한(예를 들어, 가장 복잡한) "특수 IDCT" 루틴을 선택할 것이다. 대조적으로, 본 발명의 바람직한 실시예들은 0 값들로 이루어진 로우들 및 컬럼들의 부분들(예를 들어, 0 값들로 이루어진, 각각의 1/2-로우 또는 1/2-컬럼이나 각각의 1/4-로우 또는 1/4 컬럼)에 대한 변환 동작들을 방지하거나 그러한 변환 동작들을 감소된 전력 방식으로(in a reduced-power manner) 수행하는 것에 의해 성능을 향상시키고 전력 소비를 감소시킨다. 본 발명의 바람

직한 일부 실시예들은 변환될 로우 또는 컬럼에서의 각각의 개개 0 값에 대한 변환 동작들을 방지하는 것에 의해(또는 로우 또는 컬럼에서의 각각의 개개 0 값에 대한 변환 동작들을 감소된 전력 방식으로 수행하는 것에 의해) 성능을 향상시키고 전력 소비를 감소시킨다.

<16> 미국 특허출원공개 제2005/0033788호 및 미국특허 제6,799,192호에는, 거기에서 개시된 기술에 의해 실현 가능한 성능 향상 및 전력 소비 감소 이점들이, 변환될 각각의 로우 또는 컬럼의 서브세트들을 독립적으로 프로세싱하는 것에 의해 증가될 수 있다는 제안, 및 그렇게 하기 위한 방법이나 그렇게 하는 것이 가능한지의 여부에 대한 제안이 존재하지 않는다. 대조적으로, 본 발명의 바람직한 실시예들은 변환될(예를 들어, 역변환될) 각각의 로우 또는 컬럼의 상이한 서브세트들에 대해 동일한 동작들을 순차적으로 수행할 수 있는데, 여기에서, 각각의 로우 또는 컬럼의 서브세트들은 로우 또는 컬럼의 파티션을 판정하고, 그러한 실시예들에 의해 실현 가능한 성능 향상 및 전력 소비 감소 이점들은 단지 그러한 파티션 각각을 판정하는 서브세트들의 사이즈를 감소시키는 것에 의해 증가될 수 있다. 예를 들어, 본 발명의 바람직한 일부 실시예들은 변환될 8N-비트 로우 또는 컬럼 각각의 2N-비트 서브세트들에 대해 동작들의 세트들(매 로우 또는 컬럼마다 동작들의 4개 세트들)을 순차적으로 수행하여 우수한 성능 향상 및 전력 소비 감소 이점들을 실현하고, 본 발명의 바람직한 다른 실시예들은 변환된 각각의 8N-비트 로우 또는 컬럼의 N-비트 서브세트들에 대해 동작들의 세트들(매 로우 또는 컬럼마다 동작들의 8개 세트들)을 순차적으로 수행하여 훨씬 더 양호한 성능 향상 및 전력 소비 감소 이점들을 실현한다.

<17> Rohini Krishnan 등에 의한 "Design of a 2D DCT/IDCT Application Specific VLIW Processor Supporting Scaled and Sub-sampled Blocks", 16th International Conference on VLSI Design, six pages (2003)라는 명칭의 논문에서 또 하나의 전통적인 2D 변환이 설명된다. 이 논문은 완전한 데이터 블록의 축소 버전(downscaled version)(예를 들어, 8×8 블록의 짝수 로우들을 파기하는 것에 의해 발생된 8×4 블록)을 IDCT 회로에 어서트하고, 다르게는 완전한 블록을 역변환하는데 사용될 수 있었던 IDCT 회로의 일부를 바이패스하는 것에 의한 것을 포함하는, 축소된 블록을 역변환하도록 IDCT 회로를 조작하는 것을 교수한다. 이 방법은, 궁극적으로 파기될 출력 값들의 계산을 방지할 수 있지만, 최종적인 결과에 결국은 기여하지 않을 동작들을 검출하고 스킵하지는 않는다.

<18> 미국특허 제5,883,823호에서 또 다른 전통적인 2D 변환이 설명된다. 이 변환은 변환될 입력 블록의 영역들을 식별하고, 각각의 영역을 상이하게 프로세싱한다(예를 들어, IDCT는 일부 영역들의 모든 요소들에 대해 수행되고 IDCT는 다른 영역들의 0이 아닌 요소들에 대해서만 수행된다). 예를 들어, 미국특허 제5,883,823호는 (col. 10, line 53 - col. 11, line 26에서), 8×8 블록의 일 쿼드런트(quadrant)(즉, 최저 주파수 범위들에 대응되는 4×4 쿼드런트)의 (0 여부에 상관없이) 모든 요소들에 대해 "영역적인(regional)" IDCT 계산이 수행되고 8×8 블록의 나머지 3개의 4×4 쿼드런트들(즉, 좀더 높은 주파수 범위들에 대응되는 3개의 4×4 쿼드런트들) 각각의 0이 아닌 요소들에 대해서만 다른 IDCT 계산이 수행되는 IDCT 계산을 명시적으로 교수한다.

발명의 내용

해결 하고자하는 과제

<19> 그러나, 미국특허 제5,883,823호는, IDCT 계산이 0이 아닌 요소들에 대해서만 수행되어야 하는 경우를 위해 각 영역의 0이 아닌 요소들을 식별하는 방법(또는 그러한 0이 아닌 계수들을 효율적으로 식별하는 방법), 블록의 영역에서의 0이 아닌 요소들에 대해서만 IDCT 계산을 수행하는 방법, 또는 그러한 IDCT 계산을 0이 아닌 그러한 요소들에 대해서만 효율적으로(그리고 감소된 전력을 소비하는 방식으로) 수행하는 방법을 교수하거나 제안하지 않는다.

과제 해결수단

<20> 실시예들의 부류에서, 본 발명은 입력 데이터 블록들의 시퀀스의 각 블록에 대해 2D 변환(예를 들어, IDCT)을 수행하도록 구성된 시스템인데, 여기에서, 각 블록은 입력 데이터 값들의 로우들 및 컬럼들을 구비하고, 2D 변환은 로우 변환 및 컬럼 변환을 포함한다. 이 실시예들에서, 시스템은 각 블록의 모든 로우들에 대해 로우 변환을 수행하여 부분 변환된 데이터 블록(block of partially transformed data)을 발생시킨 다음, 부분 변환된 데이터 블록의 각 컬럼에 대해 컬럼 변환을 수행하는 것에 의해서나 각 블록의 모든 컬럼들에 대해 컬럼 변환을 수행하여 부분 변환된 데이터 블록을 발생시킨 다음, 부분 변환된 데이터 블록의 각 로우에 대해 로우 변환을 수행하는 것에 의해 2D 변환을 수행하도록 구성된다. 설명을 간략화하기 위해, 각 블록의 모든 로우들에 대해 로우 변환을 수행하여 부분 변환된 데이터 블록을 발생시킨 다음, 부분 변환된 데이터 블록의 각 컬럼에 대해

컬럼 변환을 수행하는 것에 의해 2D 변환을 수행하도록 구성된 부류의 실시예들을 설명할 것이다. "로우" 및 "컬럼"으로의 모든 참조들은 부류의 다른 실시예들을 설명하기 위해, 각각, "컬럼" 및 "로우"로의 참조들로 대체될 수 있다는 것이 이해될 수 있어야 한다.

<21> 여기에서, 데이터 값들의 세트(예를 들어, 로우 또는 컬럼)의 "서브세트"라는 용어는 광범위한 의미로 사용되고, 데이터 값들의 로우(또는 컬럼), 데이터 값들의 로우(또는 컬럼)의 짝수 요소들, 데이터 값들의 로우(또는 컬럼)의 홀수 요소들, 데이터 값들의 로우(또는 컬럼)에서의 매 N번째 데이터 값, (FFT 버터플라이들에 적합한) 역비트순(bit-reversed order)의 데이터 값들의 로우 또는 컬럼의 짝수 요소들, 또는 데이터 값들의 다른 서브세트를 지시할 수 있다.

<22> 로우 변환을 구현하기 위해, 언급된 부류의 시스템은 입력 데이터 블록의 각 로우의 입력 데이터 값들에 대해 (통상적으로 곱셈들 및 덧셈들을 포함하는) 변환 동작들을 수행하여 부분 변환된 데이터 블록을 발생시키도록 구성된 회로를 포함한다. 시스템은 통상적으로, 부분 변환된 데이터가 저장되는 버퍼를 포함한다. 컬럼 변환을 구현하기 위해, 시스템은 부분 변환된 데이터 블록의 각 컬럼의 데이터 값들에 대해 (통상적으로 곱셈들 및 덧셈들을 포함하는) 변환 동작들을 수행하도록 구성된 회로를 포함한다. 언급된 부류의 각 실시예는, 입력 데이터의 각 로우에 대해 로우 변환을 수행할 때, 로우의 제1 파티션을 구비하는 데이터 값들의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지(예를 들어, 0 값들로 이루어졌는지)의 여부를 판정하고, 로우의 제1 파티션의 제1 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하며, 로우의 제1 파티션의 하나 이상의 다른 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하고, 하나 이상의 0 입력 데이터 값 및 하나 이상의 0이 아닌 입력 데이터 값을 포함하는 상기 각 로우에 대해 로우 변환을 수행할 때, 하나 이상의(그리고 바람직스럽기로는 각각의) 상기 0 입력 데이터 값에 대한 하나 이상의 변환 동작이 바이패스되거나 감소된 전력 방식으로 수행되도록 구성되는데, 여기에서, 0 값이 0이 아닌 값이라면, 그러한 변환 동작은, 다르게는 전체 전력을 소비하는 방식으로 수행될 것이다(예를 들어, 다르게는 곱셈 및 덧셈 회로를 사용해 수행될 하나 이상의 곱셈 및 덧셈이 바이패스될 것이다). 하나 이상의 0 입력 데이터 값 및 하나 이상의 0이 아닌 입력 데이터 값을 포함하는 로우 각각에 대한 로우 변환을 구현할 때, 0 입력 값에 대해 변환 동작을 수행하기 위한 그러한 일부 실시예들의 회로는, 다르게는 그러한 입력 각각을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 그것의 입력들 중 하나 이상을 업데이트하지 않으면서 동작된다.

<23> 바람직스럽게도, 언급된 부류의 실시예는, 부분 변환된 데이터의 각 컬럼에 대해 컬럼 변환을 수행할 때, 컬럼의 제1 파티션을 구비하는 데이터 값들의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지(예를 들어, 0 값들로 이루어졌는지)의 여부를 판정하고, 컬럼의 제1 파티션의 제1 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하며, 컬럼의 제1 파티션의 하나 이상의 다른 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하고, 부분 변환된 데이터의 하나 이상의 0 값 및 부분 변환된 데이터의 하나 이상의 0이 아닌 값을 포함하는 상기 컬럼 각각에 대해 컬럼 변환을 수행할 때, 그것은, 상기 값이 0이 아닌 값이라면, 부분 변환된 데이터 값의 하나 이상의(그리고 바람직스럽기로는 각각의) 상기 0 값에 대해, 다르게는 전체 전력을 소비하는 방식으로 수행할 변환 동작들 중 하나 이상(예를 들어, 다르게는 곱셈 및 덧셈 회로를 사용해 수행될 하나 이상의 곱셈 및 덧셈)을 바이패스한다(또는 감소된 전력 방식으로 수행한다). 예를 들어, 하나 이상의 0 데이터 값 및 하나 이상의 0이 아닌 데이터 값을 포함하는 각 컬럼에 대한 컬럼 변환을 구현할 때, 그러한 일 실시예는, 0값이 0이 아닌 값이라면, 다르게는 수행될 컬럼의 0 값에 대한 변환 동작을 바이패스하도록 구성된다(예를 들어, 실시예는 변환 동작을 수행하기 위한 회로를 바이패스한다).

<24> 발명의 바람직한 실시예들은, 변환될 데이터의 각 블록이 전체적으로 0 값들로 이루어졌는지의 여부를 판정한다. 블록이 전체적으로 0 값들로 이루어졌다고 판정하면, 블록의 값들에 대한 변환 동작들(로우 및 컬럼 변환 동작들 모두)은 바이패스되거나 감소된 전력 방식으로 수행된다. 바람직한 이 실시예들 또한, 변환될 데이터 블록의 각 로우 또는 컬럼의 다수의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 순차적으로(예를 들어, 반복적으로) 판정한다. 그러한 실시예의 일례는, 값들(x_i)로 이루어진 데이터의 로우(또는 컬럼)를 참조할 것인데, 여기에서, i는 $0 \le i \le N-1$ 범위의 정수이고, N은 짝수 정수이며, 로우(또는 컬럼)는 데이터 값들의 제1 서브세트 및, 제1 서브세트와는 상이한, 데이터 값들의 제2 서브세트로 파티션되고, 제1 서브세트는 데이터 값들의 제3 서브세트 및 제3 서브세트와는 상이한 제4 서브세트로 파티션되며, 제2 서브세트는 데이터 값들의 제5 서브세트 및 제5 서브세트와는 상이한 제6 서브세트로 파티션된다. 예시적 실시예는, 제1 서브세트가 전체적으로 0 값들로 이루어졌는지의 여부 및 제2 서브세트가 전체적으로 0 값들로 이루어졌는지의

여부를 판정한다. 통상적으로, 로우(또는 컬럼)가 (비디오 데이터의 프레임들에 대해 DCT를 수행하는 것에 의해 발생되는) 코사인 변환 계수들로 이루어진 경우, 제1 서브세트는 저주파수 계수들(값들(x_i), 여기에서, i는 $0 \le i \le (N/2)-1$ 범위의 정수임)로 이루어지고, 제2 서브세트는 0 값들로만 이루어지는 유의 확률을 가진다(그리고, 제1 서브세트가 0 값들로만 이루어질 확률보다 제2 서브세트가 0 값들로만 이루어질 확률이 훨씬 더 높다).

<25> 예시적 실시예는, 제1 서브세트가 전체적으로 0 값들로 이루어졌다고 판정하면, 제1 서브세트의 값들에 대한 변환 동작들을 바이패스하거나 그것들을 감소된 전력 방식으로 수행한다(예를 들어, 이 동작들을 수행하기 위한 회로가 바이패스되거나, 회로는, 다르게는 그러한 입력 각각을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 그것의 입력들 중 하나 이상이 업데이트되지 않는 상태에서, 동작된다. 제2 서브세트가 전체적으로 0 값들로 이루어졌다고 판정하면, 제2 서브세트의 값들에 대한 변환 동작들은 바이패스되거나 감소된 전력 방식으로 수행된다.

<26> 제1 서브세트 및 제2 서브세트 각각이 하나 이상의 0이 아닌 값을 포함한다고 판정하면, 예시적 실시예는, 제3 서브세트, 제4 서브세트, 제5 서브세트, 및 제6 서브세트 각각이 전체적으로 0 값들로 이루어졌는지의 여부를 판정한다. 통상적인 구현에서, 제1 서브세트는 저주파수 계수들(값들(x_i), 여기에서, i는 $0 \le i \le (N/2)-1$ 범위의 정수임)로 이루어지고, 제2 서브세트는 고주파수 계수들(값들(x_i), 여기에서, i는 $N/2 \le i \le N-1$ 범위의 정수임)로 이루어지며, 제3 서브세트는 제1 서브세트의 짝수 값들(값들(x_i), 여기에서, i는 $0 \le i \le (N/2)-1$ 범위의 짝수 정수임)로 이루어지고, 제4 서브세트는 제1 서브세트의 홀수 값들(값들(x_i), 여기에서, i는 $0 \le i \le (N/2)-1$ 범위의 홀수 정수임)로 이루어지며, 제5 서브세트는 제2 서브세트의 짝수 값들로 이루어지고, 제6 서브세트는 제2 서브세트의 홀수 값들로 이루어진다. 전체적으로 0 값들로 이루어졌다고 판정되는 제3 서브세트, 제4 서브세트, 제5 서브세트, 및 제6 서브세트 각각을 위해, 그러한 서브세트의 값들에 대한 변환 동작들은 바이패스되거나 감소된 전력 방식으로 수행된다(예를 들어, 이 동작들을 수행하기 위한 회로는 바이패스되거나, 회로는, 다르게는 그러한 입력 각각을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해 그것의 입력들 중 하나 이상을 업데이트하지 않으면서, 동작된다). 하나 이상의 0이 아닌 값을 포함하는 것으로 판정되는 제3 서브세트, 제4 서브세트, 제5 서브세트, 및 제6 서브세트 각각을 위해, 변환 동작들은 그러한 서브세트의 값들에 대해 전체 전력을 소비하는 방식으로 수행된다.

<27> 상술된 예시적 실시예에 대한 변형들에서(그리고 발명의 다른 실시예들에서), 변환될 블록의 각각의 로우 또는 컬럼(또는 로우 또는 컬럼의 서브세트)를 구비하는 데이터 값들은, 로우 또는 컬럼의 파티션(또는 그것에 관한 서브세트)을 구비하는 각각의 별개 서브세트가 전체적으로 0 값들로 이루어졌는지(또는 0 값인지)의 여부에 대한 판정 이전에 또는 판정 동안에 재순서화된다. 예를 들어, 로우가 값들(x_i)로 이루어지고(여기에서, i는 $0 \le i \le N-1$ 범위의 정수이고, N은 짝수 정수임), 파티션은 로우의 제1 및 제2의 별개 서브세트들로 이루어지며, 제1 서브세트는 로우의 제3 및 제4의 별개 서브세트들로 이루어지고, 제2 서브세트는 로우의 제5 및 제6의 별개 서브세트들로 이루어지며, 제1 서브세트는 값들(x_i)로 이루어지고(여기에서, i는 $0 \le i \le (N/2)-1$ 범위의 정수임), 제2 서브세트는 고주파수 계수들로 이루어지며(값들(x_i), 여기에서, i는 $N/2 \le i \le N-1$ 범위의 정수임),

<28> 제3 서브세트는, 제1 서브세트의 재순서화된 버전의 짝수 값들인 값들(x_i, 여기에서, i는 $0 \le i \le (N/4)-1$의 범위임)로 이루어질 수 있으며,

<29> 제4 서브세트는, 제1 서브세트의 재순서화된 버전의 홀수 값들인 값들(x_i, 여기에서, i는 $N/4 \le i \le (N/2)-1$의 범위임)로 이루어질 수 있고,

<30> 제5 서브세트는, 제2 서브세트의 재순서화된 버전의 짝수 값들인 값들(x_i, 여기에서, i는 $N/2 \le i \le (3N/4)-1$의 범위임)로 이루어질 수 있으며,

<31> 제6 서브세트는, 제2 서브세트의 재순서화된 버전의 홀수 값들인 값들(x_i, 여기에서, i는 $3N/4 \le i \le N-1$의 범위임)로 이루어질 수 있다.

<32> 실시예들의 다른 부류에서, 발명은 입력 데이터 블록들의 시퀀스에 대해 2D 변환을 수행하기 위한 방법인데, 여기에서, 블록들 각각은 입력 데이터 값들의 로우들 및 컬럼들을 구비하고, 2D 변환은 로우 변환 및 컬럼 변환을 포함한다. 그 부류의 통상적인 실시예들에서, 각 블록은 비디오 데이터에 대해 DCT 변환을 수행하는 것에 의해 발생된 DCT 계수들의 블록이고, 2D 변환은 IDCT이다. 통상적인 이 실시예들에서 그리고 부류의 다른 실시예들

에서, 시퀀스의 블록들 중 다수 또는 전부(예를 들어, 적어도 사실상 모든 블록들)는 스파스 데이터의 블록들이다. 2D 변환은 입력 데이터 블록 각각의 모든 로우들에 대해 로우 변환을 수행하여 부분 변환된 데이터 블록을 발생시킨 다음, 부분 변환된 데이터 블록의 각 컬럼에 대해 컬럼 변환을 수행하는 단계들 또는 입력 데이터 블록 각각의 모든 컬럼들에 대해 컬럼 변환을 수행하여 부분 변환된 데이터 블록을 발생시킨 다음, 부분 변환된 데이터 블록의 각 로우에 대해 로우 변환을 수행하는 단계들을 포함할 수 있다. 설명을 간략화하기 위해, 2D 변환이, 입력 데이터 블록 각각의 모든 로우들에 대해 로우 변환을 수행하여 부분 변환된 데이터 블록을 발생시키는 단계 및 부분 변환된 데이터 블록의 각 컬럼에 대해 컬럼 변환을 수행하는 단계를 포함하는 부류의 실시예들을 설명할 것이다. "로우" 및 "컬럼"으로의 모든 참조들은, 그 부류의 다른 실시예들을 설명하기 위해, 각각, "컬럼" 및 "로우"로의 참조들로써 대체될 수 있다는 것을 이해할 수 있어야 한다.

<33> 언급된 부류의 일부 실시예들에서, 방법은:

<34> (a) 입력 데이터 블록들 중 하나의 로우 각각에 대해 로우 변환을 수행하는 단계로서, 상기 로우 각각의 입력 데이터 값들에 대해 변환 동작들을 수행하는 것에 의해 부분 변환된 데이터 블록을 발생시키는 것을 포함하는 단계; 및

<35> (b) 부분 변환된 데이터 블록의 컬럼 각각에 대해 컬럼 변환을 수행하는 단계로서, 상기 각 컬럼의 데이터 값들에 대해 추가적인 변환 동작들을 수행하는 것을 포함하는 단계를 포함하고,

<36> 단계 (a)는:

<37> 상기 로우 각각의 제1 파티션을 구비하는 데이터 값들의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지(예를 들어, 0 값들로 이루어졌는지)의 여부를 판정하고, 제1 파티션의 제1 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하며, 제1 파티션의 하나 이상의 다른 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계; 및

<38> 하나 이상의 0 입력 데이터 값 및 하나 이상의 0이 아닌 입력 데이터 값을 포함하는 상기 로우 각각에 대해 로우 변환을 수행할 때, 상기 0 입력 값 각각이 0이 아닌 값이라면, 다르게는 전체 전력을 소비하는 방식으로 하나 이상의(그리고 바람직스럽기로는 각각의) 상기 0 입력 데이터 값에 대해 수행될 변환 동작들 중 하나 이상을 바이패스하는(또는 감속된 전력 방식으로 수행하는) 단계(예를 들어, 다르게는 로우의 하나 이상의(그리고 바람직스럽기로는 각각의) 상기 0 입력 데이터 값에 대해 곱셈 및 덧셈 회로를 사용해 수행될 하나 이상의 곱셈 또는 덧셈을 바이패스하는 단계)를 포함한다.

<39> 예를 들어, 하나 이상의 0 입력 데이터 값 및 하나 이상의 0이 아닌 입력 데이터 값을 포함하는 각각의 로우에 대한 로우 변환을 구현중일 때, 단계 (a)는, 다르게는 제2 입력을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 제2 입력으로 어서트되는 값을 업데이트하지 않으면서, 제1 입력 및 제2 입력을 가진 곱셈 회로를 조작하여 (제1 입력으로 어서트되는 0 입력 값이 제2 입력으로 어서트되는 제2 값만큼 곱셈되는) 곱셈 동작을 수행하는 단계를 포함한다.

<40> 바람직스럽게도, 단계 (b)는:

<41> 상기 각 컬럼의 파티션을 구비하는 데이터 값들의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지(예를 들어, 0 값들로 이루어졌는지)의 여부를 판정하고, 파티션의 제1 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하며, 파티션의 하나 이상의 다른 서브세트의 상이한 서브세트들 각각이 하나 이상의 0 값을 포함하는지의 여부를 판정하는 단계; 및

<42> 하나 이상의 0 입력 데이터 값 및 하나 이상의 0이 아닌 입력 데이터 값을 포함하는 상기 각 컬럼에 대해 컬럼 변환을 수행할 때, 각각의 상기 0 입력 값이 0이 아닌 값이라면, 다르게는 전체 전력을 소비하는 방식으로 하나 이상의(그리고 바람직스럽기로는 각각의) 상기 0 입력 데이터 값에 대해 수행될 추가적인 변환 동작들 중 하나 이상을 바이패스하는 단계(또는 감소된 전력 방식으로 수행하는 단계)(예를 들어, 다르게는 컬럼의 하나 이상의(그리고 바람직스럽기로는 각각의) 상기 0 입력 데이터 값에 대해 곱셈 및 덧셈 회로를 사용해 수행될 하나 이상의 곱셈 또는 덧셈을 바이패스하는 단계)를 포함한다.

<43> 일부 실시예들에서, 본 발명에 따른 시스템은, 본 발명에 따라 구현된 변환 엔진을 포함하는 비디오 프로세싱 시스템(예를 들어, 파이프라인형 비디오 디코딩 시스템)이다. 그러한 일부 실시예들에서, 비디오 프로세싱 시스템은 휴대용 미디어 플레이어의 비디오 프로세싱 서브시스템으로서 동작 가능하도록 구성된다. 다른 실시예들에서, 본 발명에 따른 시스템은, 본 발명에 따라 구현된 변환 엔진을 포함하는 비디오 프로세싱 서브시스템을

포함하는 휴대용 미디어 플레이어이다.

<44> 본 발명의 다른 태양들은 본 발명에 따른 시스템의 임의 실시예에서 사용하기 위한 변환 엔진들과 변환 엔진 회로 및 발명에 따른 시스템의 임의 실시예의 동작 동안 수행되는 방법들이다.

효 과

<45> 본 발명의 통상적인 실시예들에 따라 구현되는 변환 회로의 이점들은:

<46> 중복되거나 아니면 불필요한 계산들 또는 계산 단계들의 방지로 인해 (동작의 좀더 낮은 빈도(lower frequency of operation)를 허용하는) 향상된(그리고 바람직스럽기로는 최적화된) 계산 효율성(예를 들어, 입력 데이터 값들의 블록들을 변환함에 있어서, 본 발명에 따른 변환 엔진의 통상적인 실시예들의 성능은 매 블록당 0이 아닌 입력 데이터 값들의 수와 직접적으로 비례한다. 이것은, 입력 데이터 값들의 블록들을 변환함에 있어서, 매 블록당 하나 이상의 0이 아닌 값을 포함하는 로우들의 수와 실행이 비례하는 전통적인 변환 엔진들을 위한 것보다 나은 레벨 입도에 있게 된다);

<47> 전력 소비를 감소시키는 감소된 스위칭 활동;

<48> (바람직한 실시예들에서의) 지능형 중간 버퍼 메모리 관리의 제공; 및

<49> 중복되거나 아니면 불필요한 계산들 또는 계산 단계들을 방지하는 것(예를 들어, 불필요한 곱셈기 입력 토글링을 방지하는 것)에 의한 정수 변환들의 좀더 빠른 실행을 포함한다.

발명의 실시를 위한 구체적인 내용

<50> 본 발명에 따른 시스템의 실시예들이 도 1 내지 도 6을 참조하여 설명될 것이다.

<51> 도 1은, 도시된 바와 같이 시스템 버스를 따라 접속된, 비트스트림 엔진(1), (본 발명에 따라 구현될 수 있는) 변환 엔진(3), 및 픽셀 재구성 엔진(5)을 포함하는 파이프라인형 비디오 디코딩 시스템의 시스템 레벨 블록도이다. 비트스트림 엔진(1)은, 주파수 도메인 계수들의 프로세싱된 블록들을 구비하는 인코딩된 비디오 데이터를 수신하고, 상기 주파수 도메인 계수들은 비디오 픽셀들의 블록들에 대해 2D 변환(예를 들어, 2차원 DCT)을 수행하는 것에 의해 발생된다. 비트스트림 엔진(1)은, 통상적으로 그것에 대해 언패킹(unpacking) 및 역 양자화(inverse quantization)를 수행하는 것에 의해, 인코딩된 비디오 데이터를 프로세싱하고, 프로세싱된 주파수 도메인 계수들의 블록들의 결과적인 시퀀스를 변환 엔진(3)으로 어서트한다. 변환 엔진(3)은 시퀀스의 블록 각각에 대해 2D 역변환을 수행하여, 부분적으로 디코딩된 비디오 픽셀들의 블록들을 발생시키고, 부분적으로 디코딩된 비디오 픽셀들을 추가적인 비디오 디코딩을 위해 픽셀 재구성 엔진(5)으로 어서트한다. 픽셀 재구성 엔진(5)은, 부분적으로 디코딩된 비디오 픽셀들에 응답하여, 완전히 디코딩된 픽셀들을 발생시키고, 완전히 디코딩된 픽셀들을 시스템 버스로 어서트한다.

<52> 도 1의 시스템은 비디오 디코딩 능력을 갖춘 휴대용 미디어 플레이어 또는 다른 휴대용 데이터 프로세싱 시스템(또는 휴대용일 것이 의도되지 않는 데이터 프로세싱 시스템)의 서브시스템일 수 있고, 그러한 서브시스템은 본 발명에 따른 변환 엔진(3)을 구현하는 것에 의해 본 발명에 따라 구현될 수 있다. 그러한 프로세싱 시스템은 도 1에 도시되지 않은 추가 요소들을 포함할 수 있고 통상적으로 포함할 것이다.

<53> 본 발명의 일부 실시예들은, 도 1에 도시된 구조를 갖지 않는 시스템들에 의해 또는 도 2에 도시된 구조를 갖지 않는 프로세싱 시스템들에 의해 구현된다는 것이 예상된다.

<54> 도 2는 도 1의 변환 엔진(3)의 통상적인 구현 블록도 또는 데이터 값들의 블록들에 대해 2차원 변환(예를 들어, 2D 역변환)을 수행하기 위한 다른 장치의 블록도이다. 도 2는, 도시된 바와 같이 접속된, 로우/컬럼 변환 회로(7), 저장 버퍼(9), 및 컬럼/로우 변환 회로(11)를 포함한다. 일부 구현들에서, 도 2의 회로(7)는 수신된 입력 데이터의 블록들의 시퀀스의 각 블록의 모든 로우들에 대해 로우 변환을 수행함으로써 결과적인 데이터를 저장 버퍼(9)로 어서트하도록 구성되고, 회로(11)는 버퍼(9)로부터 그러한 데이터의 컬럼들을 판독하고 로우 변환으로부터 획득되는 데이터의 각 블록의 모든 컬럼들에 대해 컬럼 변환을 수행하도록 구성된다. 다른 구현들에서, 도 2의 회로(7)는 수신된 입력 데이터의 블록들의 시퀀스의 각 블록의 모든 컬럼들에 대해 컬럼 변환을 수행함으로써 결과적인 데이터를 저장 버퍼(9)로 어서트하도록 구성되고, 회로(11)는 버퍼(9)로부터 그러한 데이터의 로우들을 판독하고 컬럼 변환으로부터 획득되는 데이터의 각 블록의 모든 로우들에 대해 로우 변환을 수행하도록 구성된다.

<55> 도 3은, 도 2 구현의 회로(7)로 입력될 수 있는 유형의 ("I"으로 레이블링된) 입력 데이터 블록, (예를 들어, 도 2의 그러한 구현의 회로(7)에서) 로우 변환을 경험한 블록 I의 (블록 "Ⅱ"로 레이블링된) 버전, 및 (예를 들어, 도 2의 그러한 구현의 회로(7) 및 회로(11)에서) 로우 변환 및 컬럼 변환을 경험한 블록 I의 (블록 "Ⅲ"로 레이블링된) 버전의 도면이다. 도 3의 블록 I은, 계수들의 4개 로우들 및 4개 컬럼들을 구비하는, 주파수 도메인 계수들의 4×4 블록이고, 각각의 계수는 "0"으로 레이블링된 0 값을 갖고 "X"로 레이블링된 0이 아닌 값을 가진다. 로우들은 참조 심볼들("R1" 내지 "R4")에 의해 식별되고, 컬럼들은 참조 심볼들("a" 내지 "d")에 의해 식별된다.

<56> 통상적이고 전통적인 변환 엔진은, 전체적으로 0 값들로 이루어진 (변환될 각 블록의) 완전한 로우들 또는 컬럼들(즉, "0-로우들" 또는 "0-컬럼들")을 식별하고, 다르게는 식별된 각각의 0-로우 또는 0-컬럼에 대해 수행될 통상 변환 동작들을 바이패스하도록 구성된다. 그러한 전통적인 변환 엔진은 블록 I의 로우들(R1 및 R3)을 0-로우들로 식별할 것이고, 다르게는 그것이 로우들(R1 및 R3)에 대해 수행할 변환 계산들을 바이패스할 것이다.

<57> 그에 비해, 본 발명에 따른 변환 엔진의 통상적인 실시예(예를 들어, 발명을 구체화하는 도 2 엔진의 구현)는 변환될 각 블록의 0-로우들 및 0-컬럼들(예를 들어, 블록 I의 로우들(R1 및 R3))을 식별하고, 다르게는(그것이 로우를 0-로우로 식별하지 않았다면) 그것이 0-로우에 대해 수행할 로우 변환 계산들을 바이패스하고, 다르게는 (그것이 컬럼을 0-컬럼으로 식별하지 않았다면) 그것이 0-컬럼에 대해 수행할 컬럼 변환 계산들을 바이패스할 뿐만 아니라, 0이 아닌 로우들 및 0이 아닌 컬럼들에 대한 변환 동작들(예를 들어, 블록 I의 로우들(R2 및 R4)에 대한 로우 변환 계산들) 동안 추가 전력을 보존하도록 구성된다. 추가 전력 보존은, 전체적으로 0 데이터 값들로 이루어진 로우들의 서브세트들(및 컬럼들의 서브세트들)(예를 들어, 컬럼들(c 및 d)의 계수들로 이루어진 블록 I의 로우(R2)의 서브세트 또는 컬럼들(b 및 d)의 계수들로 이루어진 블록 I의 로우(R4)의 서브세트)을 식별하고, 다르게는 서브세트가 전체적으로 0 데이터 값들로 이루어지지 않았다면, 그러한 서브세트 각각에 대해 수행될 변환 계산들을 방지하는 것(또는 감소된 전력 방식으로 수행하는 것)에 의해 실현된다.

<58> 발명에 따른 변환 엔진의 그러한 실시예가 도 2에 도시된 구조를 가지며 DCT 변환 계수들의 4×4 블록들의 시퀀스의 블록 각각에 대해 IDCT를 수행하도록 구성되는 경우를 생각한다. 그러한 블록 각각의 로우 각각은 2개의 저주파수 계수들(예를 들어, 도 3에서의 블록 각각의 컬럼들("a" 및 "b")에서의 계수들) 및 2개의 고주파수 계수들(예를 들어, 도 3에서의 블록 각각의 컬럼들("c" 및 "d")에서의 계수들)로 이루어진다. 그러한 블록의 컬럼 각각은 2개의 저주파수 계수들(예를 들어, 도 3에서의 블록 각각의 로우들("R1" 및 "R2")에서의 계수들) 및 2개의 고주파수 계수들(예를 들어, 도 3에서의 블록 각각의 로우들("R3" 및 "R4")에서의 계수들)로 이루어진다. 이 경우, 각 로우 및 각 컬럼의 좀더 높은 주파수 계수들은, 좀더 낮은 주파수 계수들이 0 값들로만 이루어질 확률보다 0 값들로만 이루어질 확률이 높다.

<59> 본 발명에 따른 변환 엔진의 그러한 일 실시예의 회로(7)는, DCT 변환 계수들의 각 블록의 로우 각각의 2개의 최저 주파수 계수들이 전체적으로 0 값들로 이루어졌는지의 여부를 판정하고, 그러한 로우 각각의 2개의 최고 주파수 계수들이 전체적으로 0 값들로 이루어졌는지의 여부를 판정하며, 그것들이 전체적으로 0 값들로 이루어졌다면 로우 각각의 2개의 최저 주파수 계수들에 대한 로우 변환 계산들을 바이패스하거나(이 경우, 계수들이 전체적으로 0 값들로 이루어지지 않았다면, 그러한 계산들은 다르게는 이 계수들에 대해 전체 전력을 소비하는 방식으로 수행될 것임) 그러한 계산들을 감소된 전력 방식으로 수행하고, 계수들이 전체적으로 0 값들로 이루어졌다면 로우 각각의 2개의 최고 주파수 계수들에 대한 로우 변환 계산들을 바이패스하거나(이 경우, 그것들이 전체적으로 0 값들로 이루어지지 않았다면, 그러한 계산들은 다르게는 이 계수들에 대해 전체 전력을 소비하는 방식으로 수행될 것임) 그러한 계산들을 감소된 전력 방식으로 수행하도록 구성된다. 예를 들어, 그러한 실시예의 회로(7)는 블록 I의 로우들(R1, R2, R3, 및 R4) 각각의 2개의 최고 주파수 계수들에 대한 로우 변환 계산들을 바이패스하도록 구성될 수 있다.

<60> 또한, 본 발명에 따른 변환 엔진의 그러한 일 실시예의 회로(7)는 그것으로 어서트된 DCT 변환 계수들의 각 블록의 로우 각각의 2개의 최저 주파수 컴포넌트들의 짝수 계수(예를 들어, 도 3의 블록 I의 컬럼("b")에서의 각 계수)가 0 값인지의 여부를 판정하고, 그것으로 어서트된 DCT 변환 계수들의 각 블록의 로우 각각의 2개의 최저 주파수 컴포넌트들의 홀수 계수(예를 들어, 도 3의 블록 I의 컬럼("a")에서의 각 계수)가 0 값인지의 여부를 판정하며, 그것으로 어서트된 DCT 변환 계수들의 각 블록의 로우 각각의 2개의 최고 주파수 컴포넌트들의 짝수 계수(예를 들어, 도 3의 블록 I의 컬럼("d")에서의 각 계수)가 0 값인지의 여부를 판정하고, 그것으로 어서트된 DCT 변환 계수들의 각 블록의 로우 각각의 2개의 최고 주파수 컴포넌트들의 홀수 계수(예를 들어, 도 3의 블록 I의 컬럼("c")에서의 각 계수)가 0 값인지의 여부를 판정하도록 구성된다. 그러한 실시예들의 회로(7) 또한, 0 값인 것으로 판정된 그러한 계수 각각에 대해, 다르게는 (좀더 많은 전력을 소비하는 방식으로) 수행될 곱셈 계

산을 감소된 전력 방식으로 수행하여 로우를 변환하도록 구성된다. 통상적으로, 그러한 감소된 전력 곱셈 각각은, 제1 입력 및 제2 입력을 가진 곱셈 회로를 제2 입력으로 어서트된 값을 업데이트하지 않으면서 (제1 입력으로 어서트된) 0 값을 (제2 입력으로 어서트된) 제2 값만큼 곱셈하도록 동작시킴으로써, 제1 입력으로 어서트된 값이 0이 아닌 값인 경우(그리고 제2 입력에서의 업데이트된 제2 값이 그러한 0이 아닌 값으로 곱셈되어야 하는 경우), 다르게는 제2 입력을 토글링하는데 소비될 전력의 소비를 방지하는 것에 의해 수행된다.

<61> 또한, 발명에 따른 변환 엔진의 그러한 실시예의 회로(11)는 바람직스럽게도, 버퍼(9)로부터 판독된 부분 변환된 계수들의 4×4 블록 각각의 각 컬럼의 제1 절반(즉, 처음의 2개 데이터 값들)이 전체적으로 0 값들로 이루어졌는지의 여부를 판정하고, 그러한 컬럼 각각의 제2 절반(즉, 마지막 2개 데이터 값들)이 전체적으로 0 값들로 이루어졌는지의 여부를 판정하며, 그것이 전체적으로 0 값들로 이루어졌다면 각 컬럼의 제1 절반에 대한 컬럼 변환 계산들을 바이패스하거나(여기에서, 그것들이 전체적으로 0 값들로 이루어지지 않았다면, 그러한 계산들은, 다르게는 이 값들에 대해 전체 전력을 소비하는 방식으로 수행될 것임) 그러한 계산들을 감소된 전력 방식으로 수행하고, 그것이 전체적으로 0 값들로 이루어졌다면 각 컬럼의 제2 절반에 대한 컬럼 변환 계산들을 바이패스하거나(여기에서, 그것들이 전체적으로 0 값들로 이루어지지 않았다면, 그러한 계산들은, 다르게는 이 값들에 대해 전체 전력을 소비하는 방식으로 수행될 것임) 그러한 계산들을 감소된 전력 방식으로 수행하도록 구성된다.

<62> 또한, 발명에 따른 변환 엔진의 그러한 실시예의 회로(11)는 버퍼(9)로부터 판독된 부분 변환된 계수들의 4×4 블록 각각의 각 컬럼의 제1 절반에서의 짝수 데이터 값(예를 들어, 도 3의 블록 Ⅱ의 로우(R2)에서의 각 값)이 0 값인지의 여부를 판정하고, 그러한 각 컬럼의 제1 절반에서의 홀수 데이터 값(예를 들어, 도 3의 블록 Ⅱ의 로우(R1)에서의 각 계수)이 0 값인지의 여부를 판정하며, 그러한 각 컬럼의 제2 절반에서의 짝수 데이터 값(예를 들어, 도 3의 블록 Ⅱ의 로우(R4)에서의 각 계수)이 0 값인지의 여부를 판정하고, 그러한 각 컬럼의 제2 절반에서의 홀수 데이터 값(예를 들어, 도 3의 블록 Ⅱ의 로우(R3)에서의 각 계수)이 0 값인지의 여부를 판정하도록 구성된다. 그러한 실시예들의 회로(11) 또한, 0 값인 것으로 판정된 그러한 데이터 값 각각에 대해, 다르게는 (좀더 많은 전력을 소비하는 방식으로) 수행될 곱셈 계산을 감소된 전력 방식으로 수행하여 로우를 변환하도록 구성된다. 통상적으로, 그러한 감소된 전력 곱셈 각각은 제1 입력 및 제2 입력을 가진 곱셈 회로를 제2 입력으로 어서트된 값을 업데이트하지 않으면서 (제1 입력으로 어서트된) 0 값을 (제2 입력으로 어서트된) 제2 값만큼 곱셈하도록 동작시킴으로써, 제1 입력으로 어서트된 값이 0이 아닌 값인 경우(그리고 제2 입력에서의 업데이트된 제2 값이 그러한 0이 아닌 값으로 곱셈되어야 하는 경우), 다르게는 제2 입력을 토글링하는데 소비될 전력의 소비를 방지하는 것에 의해 수행된다.

<63> 각각의 로우 및 컬럼이 8개 데이터 값들을 구비하는, 상기 일례에 대한 변형들에서, 로우(또는 컬럼)의 제1(또는 제2) 절반에서의 홀수(또는 짝수) 데이터 값이 0 값인지의 여부를 판정하는 단계 각각은 로우(또는 컬럼)의 상기 절반의 제1(또는 제2) 절반에서의 데이터 값들이 전체적으로 0 값들로 이루어졌는지의 여부를 판정하는 단계로 대체되고, 엔진은, 전체적으로 0 값들로 이루어진 1/2 로우(또는 1/2 컬럼) 각각의 1/2 각각에 대한 로우(또는 컬럼) 변환 계산들을 바이패스하거나(여기에서, 그것들이 전체적으로 0 값들로 이루어지지 않았다면, 그러한 계산들은, 다르게는 이 값들에 대해 전체 전력 방식으로 수행될 것임) 그러한 계산들을 감소된 전력 방식으로 수행하도록 구성된다. 바람직스럽게도, 엔진은 또한, 그러한 1/2 로우(또는 1/2 컬럼) 각각에서의 홀수 데이터 값이 0 값인지의 여부를 판정하고, 그러한 1/2 로우(또는 1/2 컬럼) 각각의 짝수 데이터 값이 0 값인지의 여부를 판정하며, 다르게는 관련된 로우 또는 컬럼을 변환하기 위해 0 값인 것으로 판정되는 그러한 데이터 값 각각에 대해 (좀더 많은 전력을 소비하는 방식으로) 수행될 곱셈 계산을 감소된 전력 방식으로 수행하도록 구성된다.

<64> 상술된 일례 중 하나의 변형들에서, 회로(7 또는 11)에 의해 변환될 블록의 각각의 로우 또는 컬럼(또는 로우나 컬럼의 서브세트)을 구비하는 데이터 값들은, 로우 또는 컬럼(또는 그것에 관한 서브세트)의 파티션을 구비하는 각각의 별개 서브세트가 전체적으로 0 값들로 이루어졌는지(또는 0 값인지)의 여부에 대한 판정 이전에 또는 판정 동안 재순서화된다. 예를 들어, 로우(또는 컬럼)는 값들(x_i, 여기에서, i는 $0 \le i \le N-1$ 범위의 정수이고, N은 짝수 정수임)로 이루어지고, 파티션은 로우(컬럼)의 별개의 제1 및 제2 서브세트들로 이루어지며, 제1 서브세트는 로우(컬럼)의 별개의 제3 및 제4 서브세트들로 이루어지고, 제2 서브세트는 로우(컬럼)의 별개의 제5 및 제6 서브세트들로 이루어지며, 제1 서브세트는 제1 서브세트는 값들(x_i, 여기에서, i는 $0 \le i \le (N/2)-1$ 범위의 정수임)로 이루어지고, 제2 서브세트는 값들(x_i, 여기에서, i는 $N/2 \le i \le N-1$ 범위의 정수임)로 이루어지며,

<65> 제3 서브세트는 값들(x_i, 여기에서, i는 $0 \le i \le (N/4)-1$의 범위임)로 이루어질 수 있거나, 제1 서브세트의 짝수

값들로 이루어질 수 있거나, 제1 서브세트의 재순서화된 버전의 짝수 값들로 이루어질 수 있고,

<66> 제4 서브세트는 값들(x_i, 여기에서, i는 $N/4 \le i \le (N/2)-1$의 범위임)로 이루어질 수 있거나, 제1 서브세트의 홀수 값들로 이루어질 수 있거나, 제1 서브세트의 재순서화된 버전의 홀수 값들로 이루어질 수 있으며,

<67> 제5 서브세트는 값들(x_i, 여기에서, i는 $N/2 \le i \le (3N/4)-1$의 범위임)로 이루어질 수 있거나, 제2 서브세트의 짝수 값들로 이루어질 수 있거나, 제2 서브세트의 재순서화된 버전의 짝수 값들로 이루어질 수 있고,

<68> 제6 서브세트는 값들(x_i, 여기에서, i는 $3N/4 \le i \le N-1$의 범위임)로 이루어질 수 있거나, 제2 서브세트의 홀수 값들로 이루어질 수 있거나, 제2 서브세트의 재순서화된 버전의 홀수 값들로 이루어질 수 있다.

<69> 도 3의 블록 I의 로우(R4)를 다시 참조하면, 그러한 로우(R4)에서의 짝수 계수들은 0 값들이고 그러한 로우의 홀수 계수들은 전체적으로 0 값들로 이루어지지 않았다. 그러한 로우(R4)를 변환하기 위해, 발명에 따른 변환 엔진의 일부 실시예들은, 로우의 짝수 계수들에 대해 로우 변환 동작들을 수행하는 곱셈기들로의 입력들이 그러한 로우 변환 동작들의 수행 동안 토글링하는 것을 방지하는 것에 의해, 전력 소비를 감소시킬 것이다. 이것은, 변환 동안 곱셈기들에 의해 소비되는 스위칭 전력(switching power)을 감소시키는데(즉, 동적인 전력 소비(dynamic power consumption)를 감소시키는데) 도움이 된다.

<70> 다시 도 3을 참조하면, 블록 I의 로우(R4)에서는, 전체 로우에서의 제1 계수만이 0이 아닌(코딩된) 계수이다. 발명에 따른 변환 엔진의 바람직한 실시예들은 이 유형(즉, 제1 데이터 값만이 0이 아닌 유형)의 로우 또는 컬럼을 인식하고 그러한 로우 또는 컬럼을, 전체적으로 0이 아닌 값들로 이루어진 로우 또는 컬럼을 변환하는데 요구되는 것보다 낮은 전력을 소비하는 방식으로 그러한 로우 또는 컬럼을 변환하도록 구성된다. 예를 들어, (도 2에 도시된 구조를 가진 발명에 따른 변환 엔진의 실시예의) 회로(7)의 바람직한 구현들은, 로우의 0이 아닌 계수에만 응답하여 (전체 전력을 소비할) 로우의 모든 데이터 값들에 대한 통상 로우 변환(normal row transform)을 수행하지 않으면서, 제1 값만이 0이 아닌 입력 데이터 로우에 대해 감소된 전력 방식으로 로우 변환을 수행하고, 결과적인 부분 변환된 데이터 값들을 버퍼(9)에 저장하도록 구성된다.

<71> 발명에 따른 변환 엔진의 바람직한 실시예들은, 변환될 블록의 개개 데이터 값들이 0 값들인지의 여부를 판정하고(예를 들어, 계수 대 계수 기반으로 IDCT를 경험할 DCT 계수들의 블록의 개개 계수들에 대해 0-검출을 수행하고), 0 값들인 것으로 판정되는 모든 데이터 값들에 대해, 하나 이상의 상기 0 값을 변환하는데 이용될 하나 이상의 곱셈기로의 하나 이상의(그리고 바람직스럽기로는 각각의) 입력이 그것의 선행 값으로부터 스위칭하지 않는다는 것을 보장하는 것에 의해, 전력 소비를 감소시킨다. 이것은 엔진에서의 동적인 전력 소비를 감소시킨다.

<72> 도 2에 도시된 구조를 가진 발명에 따른 변환 엔진의 실시예의 바람직한 구현에서, 회로(7)는 그것에 의해 발생된 부분 변환된 데이터 값을 내부 버퍼(9)에 저장한다. 회로(11)는 이 데이터를 버퍼(9)로부터 판독하고 데이터의 각 컬럼(또는 로우)에 대해 컬럼(또는 로우) 변환을 수행한다. 통상적으로, 회로(11)는, 버퍼(9)에 데이터의 하나 이상의 컬럼(로우)가 존재하고 변환을 위해 이용 가능 하자마자, 컬럼(로우) 변환 계산을 수행할 수 있다. 이와 같이, 바람직한 실시예들에서, 회로(7)는, 회로(11)가 데이터의 완성 컬럼(로우)에 대해 컬럼(로우) 변환을 수행하기에 충분할 정도로 부분 변환된 데이터를 가능한 빨리 발생시키기 위해(그리고 버퍼(9)에 저장하기 위해) 그것이 수행하는 로우(컬럼) 변환 동작들을 재순서화하도록 구성된다. 그렇게 재순서화된(즉, 인터리브된) 방식으로 로우(컬럼) 변환을 수행하도록 회로(7)를 구성하는 것에 의해, 시스템에 의한 전력 소비는 (예를 들어, 내부 버퍼 저장을 위해 소비되는 전력을 최소화하는 것에 의해) 감소될 수 있다.

<73> 다른 방법으로, 도 2에 도시된 구조를 가진 발명에 따른 변환 엔진의 실시예에 따라 대칭 변환이 수행할 때, 회로(7)는, 변환의 대칭 특징(symmetric nature)을 이용하는 방식으로, 추가적인 로우 변환 계산들을 수행하기 전에 데이터의 2개 컬럼들을 발생시키기에(그리고 버퍼(9)에 저장하기에) 충분한 로우 변환 계산들을 수행하거나 추가적인 컬럼 변환 계산들을 수행하기 전에 데이터의 2개 로우들을 발생시키기에(그리고 버퍼(9)에 저장하기에) 충분한 컬럼 변환 계산들을 수행하도록 구성된다. 회로(7)의 그러한 실시예에 의해 수행되는 로우(또는 컬럼) 변환은 회로(11)가 버퍼링된 데이터의 추가적인 완성 컬럼(로우)에 대해 컬럼(로우) 변환을 수행하기에 충분할 정도로 부분 변환된 데이터를 가능한 빨리 발생시키도록(그리고 버퍼(9)에 저장하도록) 진행한다. 추가적인 로우 변환 계산들을 수행하기 전에, 회로(7)가 로우 변환 계산들을 수행하여 데이터의 1개 컬럼을 아니면 데이터의 2개 컬럼들을 출력하도록 구현할 것인지의 여부에 대한 판정은 성능과 버퍼 메모리 요구 사항들 사이의 트레이드오프(tradeoff)에 의존한다. 버퍼(9)가 부분 변환된 데이터의 1개 컬럼을 저장하기 위한 용량만을 가지면 되는 방식으로 로우 변환을 수행하도록 회로(7)가 구현되면, 버퍼(9)의 요구되는 용량을 최소화하

는 것의 이점은 통상적으로, 감소된 성능(예를 들어, 발명에 따른 변환 엔진이 데이터 값들의 완성 블록에 대해 역변환을 수행하기 위한 사이클들의 증가된 수)의 비용을 지불할 것이라는 것이 이해될 수 있어야 한다.

<74> 발명에 따른 (실제 곱셈없이 수행될 수 있는 정수 변환인) 전통적인 H264 기반 변환을 수행하기 위해, 발명에 따른 변환 엔진은 곱셈기들없이 구현될 수 있다. H264 기반 변환은 곱셈기들이 아니라 시프터들을 사용해 구현될 수 있다. 발명에 따라 변환 회로를 바이패스하는 것은 이 구현들에서조차도 변환 계산들을 가속할 수 있다(그리고 그것에 의해 소비되는 에너지를 감소시킬 수 있다).

<75> 다음에서는, 도 4의 흐름도를 참조하여 발명의 실시예들의 부류를 좀더 상세하게 설명한다. 도 4의 흐름도는 입력 데이터 블록들의 시퀀스의 블록 각각에 대해 수행되는 2D 변환(예를 들어, IDCT)에 관한 것인데, 여기에서, 각각의 블록은 입력 데이터 값들의 로우들 및 컬럼들을 구비하고, 2D 변환은 로우 변환 및 컬럼 변환을 포함한다. 발명에 따르면, 2D 변환은 각 블록의 모든 로우들에 대해 로우 변환을 수행하여 부분 변환된 데이터 블록을 발생시킨 다음, 부분 변환된 데이터 블록의 컬럼 각각에 대해 컬럼 변환을 수행하는 것에 의해서나 각 블록의 모든 컬럼들에 대해 컬럼 변환을 수행하여 부분 변환된 데이터 블록을 발생시킨 다음, 부분 변환된 데이터 블록의 로우 각각에 대해 로우 변환을 수행하는 것에 의해 수행된다. 도 4의 흐름도는 데이터 값들의 로우들에 대한 그러한 로우 변환 또는 데이터 값들의 컬럼들에 대한 그러한 컬럼 변환을 설명한다(예를 들어, 그것은 발명을 구체화하는 도 2의 회로(7) 구현에 의한 주파수 계수들의 로우들 또는 컬럼들의 변환이나 발명을 구체화하는 도 2의 회로(11) 구현에 의해 버퍼(9)로부터 판독되는 부분 변환된 데이터 값들의 로우들 또는 컬럼들의 변환을 설명한다). 간략화를 위해, 도 4의 다음 설명은, 데이터 값들의 로우들이 변환된다고 가정할 것이다. 설명에서의 "로우"로의 모든 참조들은 발명의 다른 실시예들을 설명하기 위해 "컬럼"으로의 참조들로써 교체될 수 있다는 것이 이해될 수 있어야 한다.

<76> 발명에 따라 역변환될 통상적인 블록은 (예를 들어, 역변환이 H264 기반 변환이라면) 매 로우마다 4개 계수들 또는 (예를 들어, 역변환이 주파수 계수들의 8×8 블록들에 대한 IDCT라면) 매 로우마다 8개 계수들을 가진다. 도 4는 어느 경우에도 적용 가능하다.

<77> 도 4에 도시된 단계들을 수행하기 전에, 발명의 통상적인 실시예들의 변환 엔진은, 변환될 데이터의 블록이 전체적으로 0 값들로 이루어졌는지의 여부를 판정한다. 블록이 전체적으로 0 값들로 이루어졌다고 판정할 때, 변환 엔진은 바이패스될 블록에 대한 변환 동작들(로우 및 컬럼 변환 동작들 모두)을 발생시킨다(예를 들어, 그것은 블록의 값들에 대해 실제 변환 동작들을 수행하지 않으면서 블록에 응답하여 0들을 출력한다). 변환 엔진이, 변환될 데이터 블록이 하나 이상의 0이 아닌 값을 포함한다고 판정하면, 그것은 로우 변환을 개시한다(즉, 그것은 도 4의 단계 20을 수행한다).

<78> 도 4의 흐름도에 따라 수행될 로우 변환의 개시(단계 20)시에, (단계 22에서는) 변환될 데이터 값들의 제1 로우가 버퍼 메모리로부터 판독된다. 버퍼 메모리는 (회로(7)가 도 4의 흐름도를 구현한다면) 도 2의 회로(7)의 입력에 결합된 메모리일 수 있거나 (회로(11)가 도 4의 흐름도를 구현한다면) 도 2의 버퍼 메모리(9)일 수 있다.

<79> 단계 24에서, 변환 엔진은, 현재 로우(단계 22의 최근 수행 동안 판독된 로우)가 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정한다. 그렇지 않으면, 변환 엔진은 (로우의 데이터 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 로우의 로우 변환 버전을 나타내는 0들을 출력하고(단계 25) 버퍼 메모리로부터 후속 로우를 판독한다(단계 22의 후속 실행). 현재 로우가 하나 이상의 0이 아닌 값을 포함하면, 엔진은 (단계 26에서), 현재 로우의 제1 절반(예를 들어, 4개 데이터 값들로 이루어진 로우의 첫번째 2개 데이터 값들)이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정한다.

<80> 현재 로우의 제1 절반이 0이 아닌 값을 하나도 포함하지 않으면, 엔진은 (로우의 제1 절반의 데이터 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 로우의 제1 절반의 로우 변환 버전을 나타내는 0들을 출력하고(단계 28), (단계 40에서) 현재 로우의 제2 절반(예를 들어, 4개 데이터 값들로 이루어진 로우의 두번째 2개 데이터 값들)이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정한다.

<81> 현재 로우의 제1 절반이 하나 이상의 0이 아닌 값을 포함하면, 엔진은 (단계 30에서), 현재 로우의 제1 절반 중 제1 절반이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정하고 (단계 32에서) 현재 로우의 제1 절반 중 제2 절반이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정한다.

<82> 현재 로우의 제1 절반 중 제1 절반이 하나 이상의 0이 아닌 값을 포함하면(예를 들어, 그것이 하나의 0이 아닌 값으로 이루어지거나 하나 이상의 0이 아닌 값을 포함하는 2개 값들로 이루어지면), 엔진은 (단계 36에서) 로우의 제1 절반 중 제1 절반의 데이터 값들에 대해 로우 변환 계산들을 수행하고 (단계 39에서) 결과적인 변환 데

이터를 버퍼 메모리에 저장한다. 단계 39를 수행하는데 이용되는 버퍼 메모리는 (회로(11)가 도 4의 흐름도를 구현하면) 도 2의 회로(11)의 출력에 결합된 버퍼 메모리 또는 (회로(7)가 도 4의 흐름도를 구현하면) 도 2의 버퍼 메모리(9)일 수 있다.

<83> 현재 로우의 제1 절반 중 제1 절반이 0이 아닌 값을 포함하지 않으면, 엔진은 단계 37을 수행하는데, 단계 37에서, 그것은:

<84> (a) (현재 로우의 제1 절반 중 제1 절반의 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제1 절반 중 제1 절반의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력한다. 그 다음, 그러한 0 값 각각은 (단계 39에서) 버퍼 메모리에 저장된다;

<85> (b) 현재 로우의 제1 절반 중 제1 절반의 데이터 값(들)에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생된 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 출력 값 각각은 (단계 39에서) 버퍼 메모리에 저장된다. 예를 들어, 엔진은 하나 이상의 상수가 어서트되는 (하나 이상의 입력을 구비하는) 제1 입력 세트 및 현재 로우의 제1 절반 중 제1 절반의 데이터 값 각각이 어서트되는 (하나 이상의 입력을 구비하는) 제2 입력 세트를 가진 곱셈 회로를, 다르게는 곱셈 회로의 관련 입력(들)을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 제1 입력 세트 및 제2 입력 세트 중 하나 이상으로 어서트되는 값을 업데이트하지 않으면서, 적어도 현재 로우의 제1 절반 중 제1 절반의 데이터 값 각각에 대해 곱셈 동작을 수행하도록 동작시키는 것에 의해 이것을 수행할 수 있다. 엔진이 IDCT를 수행중인 경우, 제1 입력 세트로 어서트되는 상수 각각은 코사인 상수이고, 일 구현에서, 제2 입력 세트로 어서트되는 데이터 값이 0 값일 때마다, 엔진은 상기 0 값과 곱셈될 코사인 상수가 업데이트되는 것(그것의 선행 값으로부터 변경되는 것)을 방지하는데, 곱셈의 결과는 코사인 상수의 값에 상관없이 0일 것이기 때문이다. 다른 구현에서, 엔진은 (0 데이터 값과 곱셈될) 제1 입력 세트로 어서트되는 코사인 상수의 업데이트를 방지하고, (곱셈기 입력 중 어느 한 쪽의 토글링을 방지하기 위해) 제2 입력 세트의 대응되는 입력으로 어서트되는 데이터 값의 업데이트를 방지하며, (이들 2개의 피연산자들을 실제로 함께 곱셈하지 않으면서) 0 데이터 값을 코사인 상수와 곱셈하는 결과를 나타내는 0 값을 어서트한다(예를 들어, 0 값을 프로세싱 파이프라인의 출력으로 멀티플렉싱한다); 또는

<86> (c) (그러한 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제1 절반 중 제1 절반에서의 하나 이상의 값의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력하고, 현재 로우의 제1 절반 중 제1 절반의 하나 이상의 다른 데이터 값에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생되는 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 0 값 및 출력 값 각각은 (단계 39에서) 버퍼 메모리에 저장된다.

<87> 현재 로우의 제1 절반 중 제2 절반이 하나 이상의 0이 아닌 값을 포함하면(예를 들어, 그것이 하나의 0이 아닌 값으로 이루어지거나 하나 이상의 0이 아닌 값을 포함하는 2개 값들로 이루어지면), 엔진은 (단계 34에서) 로우의 제1 절반 중 제2 절반의 데이터 값들에 대해 로우 변환 계산들을 수행하고 (단계 39에서) 결과적인 변환 데이터를 버퍼 메모리에 저장한다.

<88> 현재 로우의 제1 절반 중 제2 절반이 0이 아닌 값을 포함하지 않으면, 엔진은 단계 35를 수행하는데, 단계 35에서, 그것은:

<89> (a) (현재 로우의 제1 절반 중 제2 절반의 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제1 절반 중 제2 절반의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력한다. 그 다음, 그러한 0 값 각각은 (단계 39에서) 버퍼 메모리에 저장된다;

<90> (b) 현재 로우의 제1 절반 중 제2 절반의 데이터 값(들)에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생된 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 출력 값 각각은 (단계 39에서) 버퍼 메모리에 저장된다. 예를 들어, 엔진은 하나 이상의 상수가 어서트되는 (하나 이상의 입력을 구비하는) 제1 입력 세트 및 현재 로우의 제1 절반 중 제2 절반의 데이터 값 각각이 어서트되는 (하나 이상의 입력을 구비하는) 제2 입력 세트를 가진 곱셈 회로를, 다르게는 곱셈 회로의 관련 입력(들)을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 제1 입력 세트 및 제2 입력 세트 중 하나 이상으로 어서트되는 값을 업데이트하지 않으면서, 적어도 현재 로우의 제1 절반 중 제2 절반의 데이터 값 각각에 대해 곱셈 동작을 수행하도록 동작시키는 것에 의해 이것을 수행할 수 있다. 엔진이 IDCT를 수행중인 경우, 제1 입력 세트로 어서트되는 상수 각각은 코사인 상수이고, 일 구현에서, 제2 입력 세트로 어서트되는 데이터 값이 0 값일 때마다, 엔진은 상기 0 값과 곱셈될 코사인 상수가 업데이트되는 것(그것의 선행 값으로부터 변경되는 것)을 방지하는데, 곱셈의

결과는 코사인 상수의 값에 상관없이 0일 것이기 때문이다. 다른 구현에서, 엔진은 (0 데이터 값과 곱셈될) 제1 입력 세트로 어서트되는 코사인 상수의 업데이트를 방지하고, (곱셈기 입력 중 어느 한 쪽의 토글링을 방지하기 위해) 제2 입력 세트의 대응되는 입력으로 어서트되는 데이터 값의 업데이트를 방지하며, (이들 2개의 피연산자들을 실제로 함께 곱셈하지 않으면서) 0 데이터 값을 코사인 상수와 곱셈하는 결과를 나타내는 0 값을 어서트한다(예를 들어, 0 값을 프로세싱 파이프라인의 출력으로 멀티플렉싱한다); 또는

<91> (c) (그러한 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제1 절반 중 제2 절반에서의 하나 이상의 값의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력하고, 현재 로우의 제1 절반 중 제2 절반의 하나 이상의 다른 데이터 값에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생되는 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 0 값 및 출력 값 각각은 (단계 39에서) 버퍼 메모리에 저장된다.

<92> 단계들(34 및 36), 단계들(34 및 37), 단계들(35 및 36), 또는 단계들(35 및 37)에서 발생되거나 어서트된 모든 데이터 값들이 (단계 39에서) 버퍼 메모리에 저장된 후, 엔진은 (단계 40에서), 현재 로우의 제2 절반이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정한다. 현재 로우의 제2 절반이 0이 아닌 값을 하나도 포함하지 않으면, 엔진은 (로우의 상기 제2 절반의 데이터 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 로우의 제2 절반의 로우 변환 버전을 나타내는 0들을 출력하고(단계 41), 변환될 후속 로우를 버퍼 메모리로부터 판독한다(단계 22의 또 다른 실행).

<93> 현재 로우의 제2 절반이 하나 이상의 0이 아닌 값을 포함하면, 엔진은 (단계 42에서) 현재 로우의 제2 절반 중 제1 절반이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정하고 (단계 44에서) 현재 로우의 제2 절반 중 제2 절반이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정한다.

<94> 현재 로우의 제2 절반 중 제1 절반이 하나 이상의 0이 아닌 값을 포함하면(예를 들어, 그것이 하나의 0이 아닌 값으로 이루어지거나 하나 이상의 0이 아닌 값을 포함하는 2개 값들로 이루어지면), 엔진은 (단계 48에서) 로우의 제2 절반 중 제1 절반의 데이터 값들에 대해 로우 변환 계산들을 수행하고 (단계 50에서) 결과적인 변환 데이터를 버퍼 메모리에 저장한다. 단계 50을 수행하는데 이용되는 버퍼 메모리는 (회로(11)가 도 4의 흐름도를 구현하면) 도 2의 회로(11)의 출력에 결합된 버퍼 메모리 또는 (회로(7)가 도 4의 흐름도를 구현하면) 도 2의 버퍼 메모리(9)일 수 있다.

<95> 현재 로우의 제2 절반 중 제1 절반이 0이 아닌 값을 포함하지 않으면, 엔진은 단계 49를 수행하는데, 단계 49에서, 그것은:

<96> (a) (현재 로우의 제2 절반 중 제1 절반의 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제2 절반 중 제1 절반의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력한다. 그 다음, 그러한 0 값 각각은 (단계 50에서) 버퍼 메모리에 저장된다;

<97> (b) 현재 로우의 제2 절반 중 제1 절반의 데이터 값(들)에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생된 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 출력 값 각각은 (단계 50에서) 버퍼 메모리에 저장된다. 예를 들어, 엔진은 하나 이상의 상수가 어서트되는 (하나 이상의 입력을 구비하는) 제1 입력 세트 및 현재 로우의 제1 절반 중 제1 절반의 데이터 값 각각이 어서트되는 (하나 이상의 입력을 구비하는) 제2 입력 세트를 가진 곱셈 회로를, 다르게는 곱셈 회로의 관련 입력(들)을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 제1 입력 세트 및 제2 입력 세트 중 하나 이상으로 어서트되는 값을 업데이트하지 않으면서, 적어도 현재 로우의 제2 절반 중 제1 절반의 데이터 값 각각에 대한 곱셈 동작을 수행하도록 동작시키는 것에 의해 이것을 수행할 수 있다. 엔진이 IDCT를 수행중인 경우, 제1 입력 세트로 어서트되는 상수 각각은 코사인 상수이고, 일 구현에서, 제2 입력 세트로 어서트되는 데이터 값이 0 값일 때마다, 엔진은 상기 0 값과 곱셈될 코사인 상수가 업데이트되는 것(그것의 선행 값으로부터 변경되는 것)을 방지하는데, 곱셈의 결과는 코사인 상수의 값에 상관없이 0일 것이기 때문이다. 다른 구현에서, 엔진은 (0 데이터 값과 곱셈될) 제1 입력 세트로 어서트되는 코사인 상수의 업데이트를 방지하고, (곱셈기 입력 중 어느 한 쪽의 토글링을 방지하기 위해) 제2 입력 세트의 대응되는 입력으로 어서트되는 데이터 값의 업데이트를 방지하며, (이들 2개의 피연산자들을 실제로 함께 곱셈하지 않으면서) 0 데이터 값을 코사인 상수와 곱셈하는 결과를 나타내는 0 값을 어서트한다(예를 들어, 0 값을 프로세싱 파이프라인의 출력으로 멀티플렉싱한다); 또는

<98> (c) (그러한 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제2 절반 중 제1 절반에서의 하나 이상의 값의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력하고, 현재 로우의 제2

절반 중 제1 절반의 하나 이상의 다른 데이터 값에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생되는 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 0 값 및 출력 값 각각은 (단계 50에서) 버퍼 메모리에 저장된다.

<99> 현재 로우의 제2 절반 중 제2 절반이 하나 이상의 0이 아닌 값을 포함하면(예를 들어, 그것이 하나의 0이 아닌 값으로 이루어지거나 하나 이상의 0이 아닌 값을 포함하는 2개 값들로 이루어지면), 엔진은 (단계 46에서) 로우의 제2 절반 중 제2 절반의 데이터 값들에 대해 로우 변환 계산들을 수행하고 (단계 50에서) 결과적인 변환 데이터를 버퍼 메모리에 저장한다.

<100> 현재 로우의 제2 절반 중 제2 절반이 0이 아닌 값을 포함하지 않으면, 엔진은 단계 47을 수행하는데, 단계 47에서, 그것은:

<101> (a) (현재 로우의 제2 절반 중 제2 절반의 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제2 절반 중 제2 절반의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력한다. 그 다음, 그러한 0 값 각각은 (단계 50에서) 버퍼 메모리에 저장된다;

<102> (b) 현재 로우의 제2 절반 중 제2 절반의 데이터 값(들)에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생된 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 출력 값 각각은 (단계 50에서) 버퍼 메모리에 저장된다. 예를 들어, 엔진은 하나 이상의 상수가 어서트되는 (하나 이상의 입력을 구비하는) 제1 입력 세트 및 현재 로우의 제2 절반 중 제2 절반의 데이터 값 각각이 어서트되는 (하나 이상의 입력을 구비하는) 제2 입력 세트를 가진 곱셈 회로를, 다르게는 곱셈 회로의 관련 입력(들)을 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 제1 입력 세트 및 제2 입력 세트 중 하나 이상으로 어서트되는 값을 업데이트하지 않으면서, 적어도 현재 로우의 제2 절반 중 제2 절반의 데이터 값 각각에 대한 곱셈 동작을 수행하도록 동작시키는 것에 의해 이것을 수행할 수 있다. 엔진이 IDCT를 수행중인 경우, 제1 입력 세트로 어서트되는 상수 각각은 코사인 상수이고, 일 구현에서, 제2 입력 세트로 어서트되는 데이터 값이 0 값일 때마다, 엔진은 상기 0 값과 곱셈될 코사인 상수가 업데이트되는 것(그것의 선행 값으로부터 변경되는 것)을 방지하는데, 곱셈의 결과는 코사인 상수의 값에 상관없이 0일 것이기 때문이다. 다른 구현에서, 엔진은 (0 데이터 값과 곱셈될) 제1 입력 세트로 어서트되는 코사인 상수의 업데이트를 방지하고, (곱셈기 입력 중 어느 한 쪽의 토글링을 방지하기 위해) 제2 입력 세트의 대응되는 입력으로 어서트되는 데이터 값의 업데이트를 방지하며, (이들 2개의 피연산자들을 실제로 함께 곱셈하지 않으면서) 0 데이터 값을 코사인 상수와 곱셈하는 결과를 나타내는 0 값을 어서트한다(예를 들어, 0 값을 프로세싱 파이프라인의 출력으로 멀티플렉싱한다); 또는

<103> (c) (그러한 데이터 값 또는 값들에 대해 실제 로우 변환 계산들을 수행하지 않으면서) 현재 로우의 제2 절반 중 제2 절반에서의 하나 이상의 값의 로우 변환 버전을 나타내는 하나 이상의 0들을 출력하고, 현재 로우의 제2 절반 중 제2 절반의 하나 이상의 다른 데이터 값에 대해 감소된 전력 방식으로 로우 변환 계산들을 수행하는 것에 의해 발생되는 하나 이상의 데이터 값을 출력한다. 그 다음, 그러한 0 값 및 출력 값 각각은 (단계 50에서) 버퍼 메모리에 저장된다.

<104> 단계들(46 및 48), 단계들(46 및 49), 단계들(47 및 48), 또는 단계들(47 및 49)에서 발생되거나 어서트된 모든 데이터 값들이 (단계 50에서) 버퍼 메모리에 저장된 후, 엔진은 (단계 52에서) 로우 변환이 현재 블록의 모든 로우들에 대해 수행되었는지의 여부를 판정한다. 로우 변환이 현재 블록의 모든 로우들에 대해 수행되었다면, 엔진은, 그것이 로우 변환 동작들을 중단하는 상태(단계 54)로 진입한다. 그 다음, 그것은 단계 20을 다시 수행하여, 변환될 데이터 값들의 후속 블록의 프로세싱을 시작할 수 있다. 로우 변환이 현재 블록의 모든 로우들에 대해 수행되지 않았다면, 엔진은 버퍼 메모리로부터 (변환될 현재 블록의) 후속 로우를 판독한다(단계 22의 또 다른 실행).

<105> 도 5는 발명을 구체화하는 도 2의 회로(7)(또는 11)의 통상적인 구현에 이용되는 변환 회로의 블록도이다. 데이터의 4×4 블록들을 변환하도록 설계된 회로(7)(또는 11)의 구현들은 통상적으로, 변환될 각 로우 또는 컬럼의 4개 데이터 값들(I0, I1, I2, 및 I3)을 변환하기 위한, 도 5 회로의 1개 유닛을 포함할 것이다. 데이터의 8×8 블록들을 변환하도록 설계된 회로(7)(또는 11)의 구현들은 통상적으로 도 5 회로의 2개 유닛들: 하나는 변환될 각 로우 또는 컬럼의 제1 절반의 4개 데이터 값들(I0, I1, I2, 및 I3)을 변환하기 위한 유닛이고; (도 5에 도시된 상수들(C1, C2, C3, 및 C4) 대신에 코사인 상수들의 제2 세트(C4, C5, C6, 및 C7)를 사용해) 변환될 각 로우 또는 컬럼의 제2 절반의 4개 데이터 값들(I4, I5, I6, 및 I7)을 변환하기 위한 동일한 유닛을 포함할 것이다.

<106> 도 5의 곱셈 유닛들(M1, M2, M3, 및 M4) 각각은 (코사인 상수들(C1, C2, C3, 및 C4) 중 하나가 어서트되는) 제1 입력, (데이터 값들(I0, I1, I2, 및 I3) 중 하나가 어서트되는) 제2 입력, 및 그것이 2개 피연산자들의 곱을 나타내는 데이터 값을 어서트하는 출력을 포함한다. 또한, 도 5 회로는, 도시된 바와 같이 접속된, (전처리 로직(100;pre-processing logic)이라고 할) 전처리 및 제어 로직(100), 덧셈 회로들(A1, A2, A3, A4, 및 A5), 뺄셈 회로(S1), 및 (플립-플롭들로서 구현된) 지연 회로들(A 및 B)을 포함하고, 그것으로 어서트된 4개 입력 값들의 세트 각각에 대해 파이프라인형 로우 변환 또는 컬럼 변환을 수행하여, 각각의 입력 값에 응답하여 출력 값(Y0)(또는 Y1)을 발생시키도록 구성된다. 동작 동안, 지연 회로(A)는 통상적으로, 덧셈 회로(A5), 뺄셈 회로(S1), 및 덧셈 회로(A3) 중 (로직(100)에 의해 어서트되는 제어 신호들에 의해 판정되는 바에 따라) 선택된 각각으로의 덧셈 회로(A3)의 출력의 어서션을, 파이프라인형 변환의 일 사이클만큼, 지연시킨다. 동작 동안, 지연 회로(B)는 통상적으로, 덧셈 회로(A5), 뺄셈 회로(S1), 및 덧셈 회로(A4) 중 (로직(100)에 의해 어서트되는 제어 신호들에 의해 판정되는 바에 따라) 선택된 각각으로의 덧셈 회로(A4)의 출력의 어서션을, 파이프라인형 변환의 일 사이클만큼, 지연시킨다.

<107> 도 5 회로가 데이터의 4×4 블록들을 변환하도록 설계되는 회로(7)(또는 11)의 구현에 포함될 때, 전처리 로직(100)은 도 4의 0(또는 0이 아닌) 값 판정 동작들을 수행하도록 구성된다. 도 5 회로의 2개 유닛들이 데이터의 8×8 블록들을 변환하도록 설계된 회로(7)(또는 11)의 구현에 포함될 때, 유닛들(100) 각각은 변환될 데이터 값들의 각 로우 또는 컬럼의 상이한 서브세트(예를 들어, 절반)에 대해 도 4의 0(또는 0이 아닌) 값 판정 동작들을 수행하도록 구성된다. 회로(7)(또는 11)의 임의 구현에서, 전처리 로직(100)은 관련 버퍼 메모리(예를 들어, 도 2의 버퍼(9) 또는 도 2의 회로(7)의 입력에 결합된 버퍼 메모리)로부터 변환될 입력 데이터 값들을 판독하고 도 4에 의해 요구되는 나머지 동작들을 구현하는데 필요한 제어 신호들을 발생시키도록(그리고 도 5의 나머지 요소들 중 적합한 요소들로 어서트하도록) 구성된다. 예를 들어, 로직(100)이 (예를 들어, 도 4의 단계 30 동안), 계수(I0)가 0 값이라고 판정하면, 그것은 제어 신호를 어서트하여, 선행의 동작 사이클 동안 곱셈기(M1)로 어서트된 계수(C0)가, 곱셈기(M1)가 0 값 계수(I0)를 상수만큼 곱하기 위한 각각의 후속 사이클 동안, 계속해서 곱셈기(M1)로 어서트되게 할 수 있다.

<108> 통상적인 구현에서, 전처리 로직(100)은, 변환될 4개의 새로운 입력 데이터 값들의 각 세트를 (예를 들어, 버퍼(9)로부터) 수신할 때, 출력 값들(Y0 및 Y1)을 0으로 리셋한다. 이 경우, 전처리 로직(100)이, (예를 들어, 도 4의 단계(25, 28, 41, 35, 37, 47, 또는 49)에서) 입력 값들 중 하나에 응답하여 0 값(즉, Y0 또는 Y1의 0 값)이 출력되어야 한다고 판정할 때, 그것은, 이 요소들이, 다르게는 이 입력 값을 변환하도록 동작할 사이클(들) 동안 회로 요소들(M1, M2, M3, M4, A1, A2, A3, A4, A, B, A5, 및 S1)의 입력들의 모든 토글링을 방지하는 대신, Y0(또는 Y1)의 리셋된 0 값이 관련 버퍼(예를 들어, 버퍼 메모리(9) 또는 회로(11)의 출력에 결합된 버퍼 메모리)로 어서트되게 할 수도 있다.

<109> 변환될 데이터 값들의 재순서화(reordering)를 요구하는 실시예들에서, 전처리 유닛(100)은 바람직스럽게도 그러한 재순서화를 수행하도록 구성된다. 예를 들어, 도 4를 참조하여 설명된 것과 유사한 변환들에서 1/2 로우 또는 1/2 컬럼의 제1(또는 제2) 절반(또는 1/2 로우 또는 1/2 컬럼의 절반)에서의 데이터 값들이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정하는 각각의 단계를 수행하기 위해, 전처리 유닛(100)은 요구되는 판정을 수행하기 위해 어서트될 데이터 값들을 재순서화할 수 있다. 예를 들어, 유닛(100)은, 재순서화된 1/2 로우 또는 재순서화된 1/2 컬럼의 제1(또는 제2) 절반(또는 재순서화된 1/2 로우 또는 재순서화된 1/2 컬럼의 절반)에서의 데이터 값들이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정함으로써, 1/2 로우 또는 1/2 컬럼의 재순서화되지 않은 버전(또는 1/2 로우 또는 1/2 컬럼의 재순서화되지 않은 버전의 절반)에서의 짝수(또는 홀수) 데이터 값들이 하나 이상의 0이 아닌 값을 포함하는지의 여부를 판정할 수 있다. 전처리 유닛(100)의 일부 구현들은, 1/2 로우들(또는 1/2 부분 로우들(half partial rows)) 또는 1/2 컬럼들(또는 1/2 부분 컬럼들)이 아닌 로우들 또는 컬럼들(이나 부분 로우들 또는 부분 컬럼들)의 서브세트들의 재순서화를 수행할 수도 있다.

<110> 도 6은 발명에 따른 파이프라인형 비디오 디코딩 시스템의 실시예의 요소들의 블록도이다. 발명의 다른 실시예들에서의 구현을 위해 도 6에 도시된 설계에 대한 다수 변형들이 예상된다(예를 들어, 0 검출 회로(132, 134, 및 135-138)는 생략될 수 있고 그것의 기능들은 0 검출 회로(130) 또는 다른 0 검출 회로에 의해 수행될 수 있다). 도 6의 시스템은, 도시된 바와 같이 접속된, 전처리 유닛(120), 변환 엔진(122), 및 후처리 유닛(124)을 포함한다.

<111> 유닛(120)은 입력 데이터 값들(예를 들어, 비디오 픽셀들의 블록들에 대해 2차원 DCT, IDCT, 또는 다른 2D 변환을 수행하는 것에 의해 발생된 입력 주파수 도메인 계수들)의 블록들의 시퀀스를 수신하도록 결합된다. 또한,

유닛(120)은 도 6 시스템을 통한 제1 전달 이후에 후처리 유닛(124)으로부터 출력되는 (도 6에서 "제1 스테이지로부터의 출력" 및 "제1 스테이지 출력"으로서 식별되는) 데이터 값들을 수신하도록 결합됨으로써, (유닛(124)으로부터의 출력 이후 및 유닛(120)으로의 어서션 이전에 버퍼 메모리에 선택적으로 버퍼링되는) 이 데이터 값들은 (예를 들어, 도 6 회로를 통한 제 1 전달 동안 데이터 값들의 컬럼들이 컬럼 변환을 경험했다면, 데이터 값들의 로우들에 대해 로우 변환을 수행하기 위한) 도 6 회로를 통한 제2 전달 동안 프로세싱의 제2 스테이지를 경험할 수 있다.

<112> 유닛(120)의 멀티플렉서(129)는 (도 6에서의 프로세싱을 경험하지 않은) 입력 데이터 값들이나 (도 6 시스템을 통한 선행 전달에서 프로세싱을 경험한) 후처리 유닛(124;post-processing unit)으로부터 출력된 데이터 값들을 선택하고 선택된 값들을 0 검출 로직(130)에 어서트한다. 선택된 값들은 (선택적으로, 로직(130)이 그것들이 전체적으로 0 값들로 이루어졌다고 판정하는 경우를 제외하고) 0 검출 로직(130)을 통해 전달된다. 그 다음, 선택적으로, 유닛(120)내의 다른 회로는 로직(130)을 통해 전달된 값들에 대해 예비 프로세싱 동작들을 수행하여 전처리된 데이터 값들을 발생시킨다(이 동작들은, 도 6 시스템을 사용해 구현될 특정 변환을 구현할 것이 요구될 때 수행된다). 유닛(120)은 (로직(130)의 출력에 대해 예비 프로세싱이 수행되지 않는다면, 로직(130)의 출력과 동일한) 전처리된 데이터 값들을 변환 엔진(122)으로 어서트한다.

<113> 전처리 유닛(120)은 통상적으로, 그것으로 어서트된 데이터 값들의 세트 각각을, 엔진(122)이 구현될 변환을 수행하기에 적합한 서브세트들로 분리하도록 구성된다. 유닛(120)이 서브세트들로의 분리를 실현하는 방식은 구현될 변환(예를 들어, 8×8 DCT, 8×8 IDCT, 또는 8×8 Hadamard)에 의존할 것이고, 외부 유닛으로부터 유닛(120)으로 어서트되는 제어 신호들에 의해 판정될 수 있어, 서브세트 선택은 구현될 변환에 기초해 맞춤화될 수 있다.

<114> 변환 엔진(122)은, 그것이 유닛(120)으로부터 수신하는 데이터 값들의 블록 각각에 대해 2D 변환을 수행하여 변환된 데이터 값들의 블록들(예를 들어, 부분 디코딩된 비디오 픽셀들의 블록들)을 발생시키고, 변환된 데이터 값들을 선택적인 추가 프로세싱을 위해 후처리 유닛(124)으로 어서트한다. 좀더 구체적으로, 변환 엔진(122)은 유닛(120)으로부터 그것으로 어서트된 4개 데이터 값들의 각 세트에 대해 파이프라인형 로우 변환 또는 컬럼 변환을 수행하여, 이들 4개 데이터 값들 각각에 응답하여 출력 값을 발생시키도록 구성된다. 4개 데이터 값들의 그러한 세트 각각은 통상적으로 데이터 값들의 블록의 로우(또는 컬럼)이거나 데이터 값들의 블록의 로우(또는 컬럼)의 서브세트이다.

<115> 후처리 유닛(124)은 선택적으로 (즉, 도 6의 시스템에 의해 수행중인 특정 변환을 위해 적합한 때에), 엔진(122)으로부터 수신되는 변환 값들에 대해 후처리 동작들을 수행하여 출력 데이터 값들을 발생시킨다. 출력 데이터 값들은 유닛(124)으로부터 시스템 버스로 어서트될 수 있고 및/또는 유닛(120)의 하나 이상의 입력으로 (또는 그것들이 유닛(120)의 하나 이상의 입력으로 어서트되는 버퍼 메모리로) 피드백될 수 있다. 통상적으로, 유닛(124)은 라운딩(rounding), 포화(saturation), 및 (필요하다면) 최종적인 패킹을 수행한다. 유닛(124)에 의해 수행되는 특유의 후처리 동작들은, 도 6의 회로가 유닛(122)으로의 입력에서 제1 스테이지를 아니면 제2 스테이지를 구현중인지에 의존한다.

<116> 도 6의 변환 엔진(122)은, 변환될 각 로우 또는 컬럼의 4개 데이터 값들(I0, I1, I2, 및 I3)을 상수들(C0, C1, C2, 및 C3)의 세트를 사용해 변환하는 것에 의한 것을 포함하여, 데이터의 4×4 블록들을 변환하도록 설계된다. 발명의 일부 다른 실시예들에서, 데이터의 8×8 블록들을 변환하기 위한 회로는 2개의 변환 엔진들(122): 변환될 각 로우 또는 컬럼의 제1 서브세트(예를 들어, 제1 절반)의 4개 데이터 값들(I0, I1, I2, 및 I3)을 변환하기 위한 하나의 엔진; 및 (도 6의 블록(122)에 도시된 상수들(C0, C1, C2, 및 C3) 대신에 상수들(C4, C5, C6, 및 C7)의 다른 세트를 사용해) 변환될 각 로우 또는 컬럼의 제2 서브세트(예를 들어, 제2 절반)의 4개 데이터 값들(I4, I5, I6, 및 I7)을 변환하기 위한 동일한 엔진을 포함할 것이다. 이용될 상수들(예를 들어, C0, C1, C2, 및 C3)은 외부 유닛으로부터 엔진(122)으로 어서트되는 제어 신호들에 의해 판정될 수 있고, 그에 따라, 상수들의 선택은 수행될 변환에 기초해 맞춤화될 수 있다.

<117> 통상적으로, 전처리 유닛(120)의 0 검출 로직(130)은, 변환될 데이터의 전체 블록이 전체적으로 (변환될 데이터의 4×4 블록의 4개 로우들 또는 컬럼들 모두가) 0 값들로 이루어졌는지의 여부를 판정하도록 구성된다. 예를 들어, 그것은, 블록의 제1 로우(컬럼)를 다운스트림 회로로 전달하기 전에, 블록의 모든 로우들(컬럼들)에 대해 그러한 판정을 수행하기에 충분한 지연을 제공하는 시프트 레지스터를 포함할 수도 있다. 블록이 전체적으로 0 값들로 이루어졌다고 판정할 때, 로직(130)은, 다르게는 엔진(122)에 의해 블록에 대해 후속적으로 수행될 변환 동작들의 (예를 들어, 상술된 바와 같은) 바이패스를 발생시킬 뿐만 아니라 통상적으로, 블록의 값들에 대한 엔

진(122) 및 유닛(124) 중 하나 또는 양자에 의한 실제 변환 동작들의 실행없이, 블록에 응답하여 유닛(124)의 멀티플렉서(139)로부터 소정 값들(예를 들어, 0들)이 출력되게 하는 적합한 제어 비트들을 도 6 시스템의 다른 요소들로 어서트한다.

<118> 전처리 유닛(120)의 0 검출 로직(130)이 데이터의 전체 블록이 선행 문단에서 설명된 바와 같이 전체적으로 0 값들로 이루어졌는지의 여부를 판정하도록 구성되는지에 상관없이, 로직(130)은 멀티플렉서(129)로부터 로직(130)으로 어서트된 각각의 개개 로우(또는 컬럼)의 데이터 값들에 대해 0 검출을 수행하여 그러한 로우(또는 컬럼)의 모든 데이터 값들이 0 값들인지의 여부를 판정하도록 구성된다(즉, 그것은 도 4의 단계 24를 수행한다). 그러한 로우(또는 컬럼) 각각이 4개 데이터 값들로 이루어진 경우, 로직(130)은, 로우(컬럼)가 4개의 0 비트들로 이루어졌는지의 여부를 판정한다. 로직(130)이, 로우(또는 컬럼)가 전체적으로 0 비트들로 이루어졌다고 판정하면, 그것은, 다르게는 로우(컬럼)에 대해 엔진(122)에 의해 후속적으로 수행될 변환 동작들이 바이패스되게 할 뿐만 아니라 통상적으로, 로우(컬럼)의 값들에 대한 엔진(122) 및 유닛(124) 중 하나 또는 양자에 의한 실제 변환 동작들의 실행없이, 로우(컬럼)에 응답하여 유닛(124)의 멀티플렉서(139)로부터 소정 값들(예를 들어, 0들)이 출력되게 하는 적합한 제어 비트들을 어서트한다.

<119> 일부 구현들에서, 로직(130)은, 멀티플렉서(129)로부터 그것으로 어서트되는 데이터 값들의 각각의 개개 로우(또는 컬럼)의 제1 데이터 값이 0이 아닌 값이고 로우(또는 컬럼)의 다른 모든 값들은 0 값들인지의 여부를 판정한다. 처음의 0이 아닌 값을 제외하면 0 값들로 이루어진 로우(컬럼)을 식별하는 것에 응답하여, 로직(130)의 그러한 구현은, 다르게는 로우(컬럼)에 대해 엔진(122)에 의해 후속적으로 수행될 변환 동작들이 바이패스되게 할 뿐만 아니라 통상적으로, 로우(컬럼)의 값들에 대한 엔진(122) 및 유닛(124) 중 하나 또는 양자에 의한 실제 변환 동작들의 실행없이, 로우(컬럼)에 응답하여 유닛(124)의 멀티플렉서(139)로부터 소정 값들(예를 들어, 0 값들)이 출력되게 하는 적합한 제어 비트들을 어서트한다.

<120> 엔진(122)의 0 검출 로직(132)은 유닛(120)으로부터 엔진(122)으로 어서트되는 각 로우(또는 컬럼)의 제1 서브세트(예를 들어, 처음의 2개 데이터 값들인 제1 절반)에 대해 0 검출을 수행하여 그러한 제1 서브세트의 모든 데이터 값들이 0 값들인지의 여부를 판정하도록 구성된다(예를 들어, 그것은 도 4의 단계 26을 수행한다). 로직(132)이, 제1 서브세트(예를 들어, 1/2 로우 또는 1/2 컬럼)가 전체적으로 0 비트들로 이루어졌다고 판정하면, 그것은, 다르게는 제1 서브세트에 대해 엔진(122)에 의해 후속적으로 수행될 변환 동작들이 바이패스되게 할 뿐만 아니라 통상적으로, 제1 서브세트의 값들에 대한 엔진(122)에 의한 실제 변환 동작들의 실행없이, 제1 서브세트에 응답하여 엔진(122)의 멀티플렉서(140)로부터 소정 값들(예를 들어, 0들)이 출력되게 하는 적합한 제어 비트들을 어서트한다. 로직(132)이, 현재 로우(컬럼)의 제1 서브세트(예를 들어, 제1 절반)가 하나 이상의 0이 아닌 값을 포함한다고 판정하면, 0 검출 로직 유닛들(135 및 136)은 제1 서브세트의 제1 및 제2 값들에 대해 0 검출을 수행한다.

<121> 로직(135)이, 현재 로우(컬럼)의 제1 서브세트의 제1 데이터 값이 0 값이라고 판정하면, 그것은 (그것의 제1 입력으로 변환 상수(C0)가 어서트되고 그것의 제2 입력으로 현재 로우 또는 컬럼의 0 데이터 값이 어서트되는) 곱셈 회로(141)로 제어 비트를 어서트하여, 회로(141)가, 다르게는 제1 입력을 상수(C0)의 업데이트된 값으로 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 그것의 제1 입력으로 어서트된 선행 값을 업데이트하지 않으면서, 그것의 제2 입력으로 어서트된 값에 대해 곱셈 동작을 수행하게 한다.

<122> 마찬가지로, 로직(136)이, 현재 로우(컬럼)의 제1 서브세트의 제2 데이터 값이 0 값이라고 판정하면, 그것은 (그것의 제1 입력으로 변환 상수(C1)가 어서트되고 그것의 제2 입력으로 현재 로우 또는 컬럼의 그러한 0 데이터 값이 어서트되는) 곱셈 회로(142)로 제어 비트를 어서트하여, 회로(142)가, 다르게는 제1 입력을 상수(C1)의 업데이트된 값으로 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 그것의 제1 입력으로 어서트된 선행 값을 업데이트하지 않으면서, 그것의 제2 입력으로 어서트된 값에 대해 곱셈 동작을 수행하게 한다.

<123> 엔진(122)의 0 검출 로직(134)은 유닛(120)으로부터 엔진(122)으로 어서트된 각 로우(또는 컬럼)의 제2 서브세트(예를 들어, 처음의 2개 데이터 값들인 제2 절반)에 대해 0 검출을 수행하여 그러한 제2 서브세트의 모든 데이터 값들이 0 값들인지의 여부를 판정하도록 구성된다(예를 들어, 그것은 도 4의 단계 40을 수행한다). 로직(134)이, 제2 서브세트가 전체적으로 0 비트들로 이루어졌다고 판정하면, 그것은, 다르게는 제2 서브세트(예를 들어, 1/2 로우 또는 1/2 컬럼)에 대해 엔진(122)에 의해 후속적으로 수행될 변환 동작들이 바이패스되게 할 뿐만 아니라 통상적으로, 제2 서브세트의 값들에 대한 엔진(122)에 의한 실제 변환 동작들의 실행없이, 제2 서브세트에 응답하여 엔진(122)의 멀티플렉서(141)로부터 소정 값들(예를 들어, 0들)이 출력되게 하는 적합한 제어 비트들을 어서트한다. 로직(134)이 현재 로우(컬럼)의 제2 서브세트가 하나 이상의 0이 아닌 값을 포함한다고

판정하면, 0 검출 로직 유닛들(137 및 138)은 제2 서브세트의 제1 및 제2 값들에 대해 0 검출을 수행한다.

<124> 로직(137)이 현재 로우(컬럼)의 제2 서브세트의 제1 데이터 값이 0 값이라고 판정하면, 그것은 (그것의 제1 입력으로 변환 상수(C2)가 어서트되고 그것의 제2 입력으로 현재 로우 또는 컬럼의 0 데이터 값이 어서트되는) 곱셈 회로(143)로 제어 비트를 어서트하여, 회로(143)가, 다르게는 제1 입력을 상수(C2)의 업데이트된 값으로 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 그것의 제1 입력으로 어서트된 선행 값을 업데이트하지 않으면서, 그것의 제2 입력으로 어서트된 값에 대해 곱셈 동작을 수행하게 한다.

<125> 마찬가지로, 로직(138)이 현재 로우(컬럼)의 제2 서브세트의 제2 데이터 값이 0 값이라고 판정하면, 그것은 (그것의 제1 입력으로 변환 상수(C3)가 어서트되고 그것의 제2 입력으로 현재 로우 또는 컬럼의 그러한 0 데이터 값이 어서트되는) 곱셈 회로(144)로 제어 비트를 어서트하여, 회로(144)가, 다르게는 제1 입력을 상수(C3)의 업데이트된 값으로 토글링하는데 소비될 전력을 소비하는 것을 방지하기 위해, 그것의 제1 입력으로 어서트된 선행 값을 업데이트하지 않으면서, 그것의 제2 입력으로 어서트된 값에 대해 곱셈 동작을 수행하게 한다.

<126> 도 6의 멀티플렉서들(129, 140, 141, 150, 151, 152, 및 153) 각각은 동작 동안, 도 6 시스템의 파이프라인형 동작의 현재 사이클 동안 그것의 입력들로 어서트되고 있는 값들 중 하나를 그것의 출력을 통해 전달하도록 제어된다. 도 6의 회로들(156 및 157) 각각은, 동작 동안 그것의 출력에서, 도 6 시스템의 현재의 동작 사이클 동안 그것의 입력들로 어서트되고 있는 값들의 합을 어서트하도록 제어되는 덧셈 회로이다. 도 6의 회로들(154 및 155)은, 도 6 시스템에 의해 수행되고 있는 특유의 변환을 위해 적합한 레이턴시를 가진 (플립플롭들로 구현된) 지연 회로들이다.

<127> 발명의 다른 일부 실시예들은, 도 6의 논리 회로(132, 134, 135, 136, 137, 및 138)가 생략되고 생략된 회로의 기능들이 도 6의 논리 회로(130)의 변경된 버전에 의해 수행되는 것을 제외하면, 도 6과 동일한 회로로 이루어지거나 도 6과 동일한 회로를 포함한다. 구체적으로, 로직(130)의 그러한 변경 버전은 통상적으로, 변환될 데이터의 전체 블록이 전체적으로(즉, 변환될 데이터의 4×4 블록의 4개 로우들 또는 컬럼들 모두가) 0 값들로 이루어졌는지의 여부, 변환될 블록의 데이터의 각 로우(또는 컬럼)가 전체적으로 0 값들로 이루어졌는지의 여부, 및 변환될 블록의 데이터의 각각의 1/2 로우(또는 컬럼)의 데이터 값 각각이 0 값인지의 여부를 판정하도록 구성될 것이고, 그러한 판정들에 응답하여, 도 6의 로직 회로(130, 132, 134, 135, 136, 137, 및 138)에 의해 어서트되는 상술된 유형들의 제어 신호들을 (시스템의 다른 요소들로) 어서트한다.

<128> 도 6의 시스템은, 예를 들어, 비디오 디코딩 능력을 가진, 휴대용 미디어 플레이어 또는 다른 휴대용 데이터 프로세싱 시스템(이나 휴대용일 것이 의도되지 않는 데이터 프로세싱 시스템)의 서브시스템일 수 있다. 그러한 미디어 플레이어 또는 다른 프로세싱 시스템은 도 6에 도시되지 않은 추가 요소들을 포함할 수 있고 통상적으로 포함할 것이다.

<129> 상이한 다수 포맷들 중 어떤 것을 가진 데이터의 블록들에 대해 다수의 상이한 2D 변환들 중 어떤 것을 수행하기 위한 발명의 다양한 실시예들을 구현하기 위해, 도 4를 참조하여 도시되고 설명된 특유의 단계들에 대한 그리고 도 5 및 도 6을 참조하여 도시되고 설명된 특유의 구조에 대한 변형들이 구현될 것이다.

<130> 여기에서는 본 발명의 일부 실시예들이 예시되고 설명되지만, 발명은 청구항들에 의해 정의되고, 설명되고 도시된 특유의 실시예들로 제한되지 않는다는 것도 이해할 수 있어야 한다.

도면의 간단한 설명

<131> 도 1은 본 발명의 실시예들에 따라 구현될 수 있는 변환 엔진을 포함하는 파이프라인형 비디오 디코딩 시스템의 블록도이다.

<132> 도 2는 본 발명에 따라 구현될 수 있는 (비디오 디코딩 능력을 가진) 프로세싱 시스템의 블록도이다.

<133> 도 3은 입력 데이터 블록, 로우 변환을 경험한 블록 버전, 및 로우 변환과 컬럼 변환을 경험한 블록 버전의 도면이다.

<134> 도 4는 (본 발명에 따른 시스템의 실시예들의 부류에 의해 수행될 수 있는) 발명에 따른 방법의 실시예의 단계들의 흐름도이다.

<135> 도 5는 본 발명을 구체화하는 도 2의 회로(7)(또는 회로(11))의 통상적인 구현에서 이용되는 변환 회로의 블록도이다.

<136> 도 6은 본 발명에 따른 파이프라인형 비디오 디코딩 시스템의 실시예의 요소들의 블록도이다.

<137> <도면의 주요 부분에 대한 부호의 설명>

<138> 1 : 비트스트림 엔진

<139> 3 : 변환 엔진

<140> 5 : 픽셀 재구성 및 디블로킹

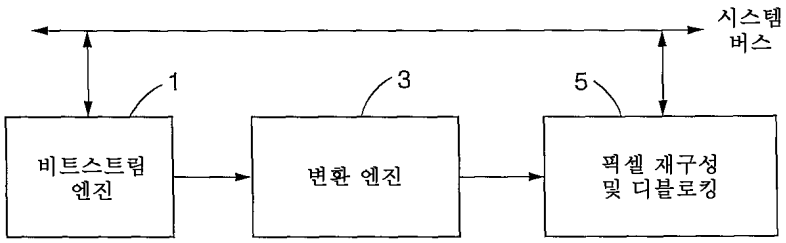
<141> 7 : 로우/컬럼 변환

<142> 9 : 중간 저장 버퍼

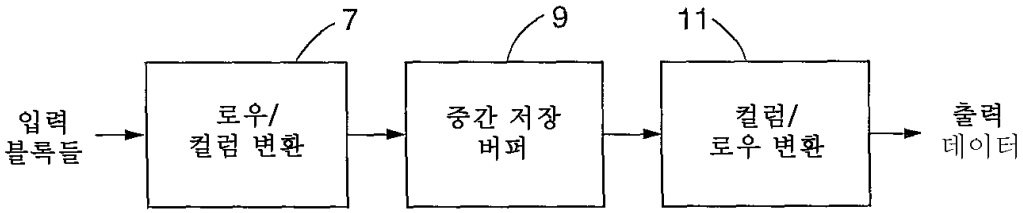
<143> 11 : 컬럼/로우 변환

도면

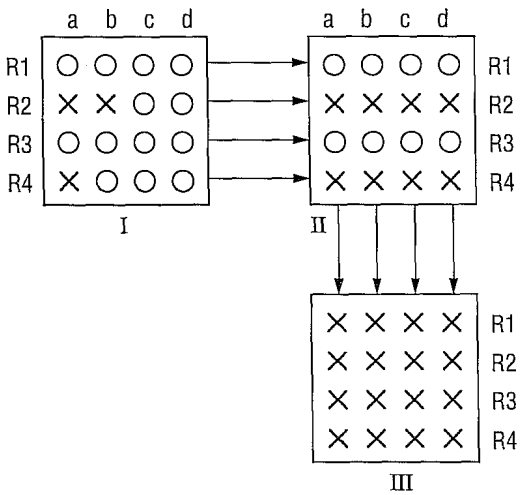
도면1



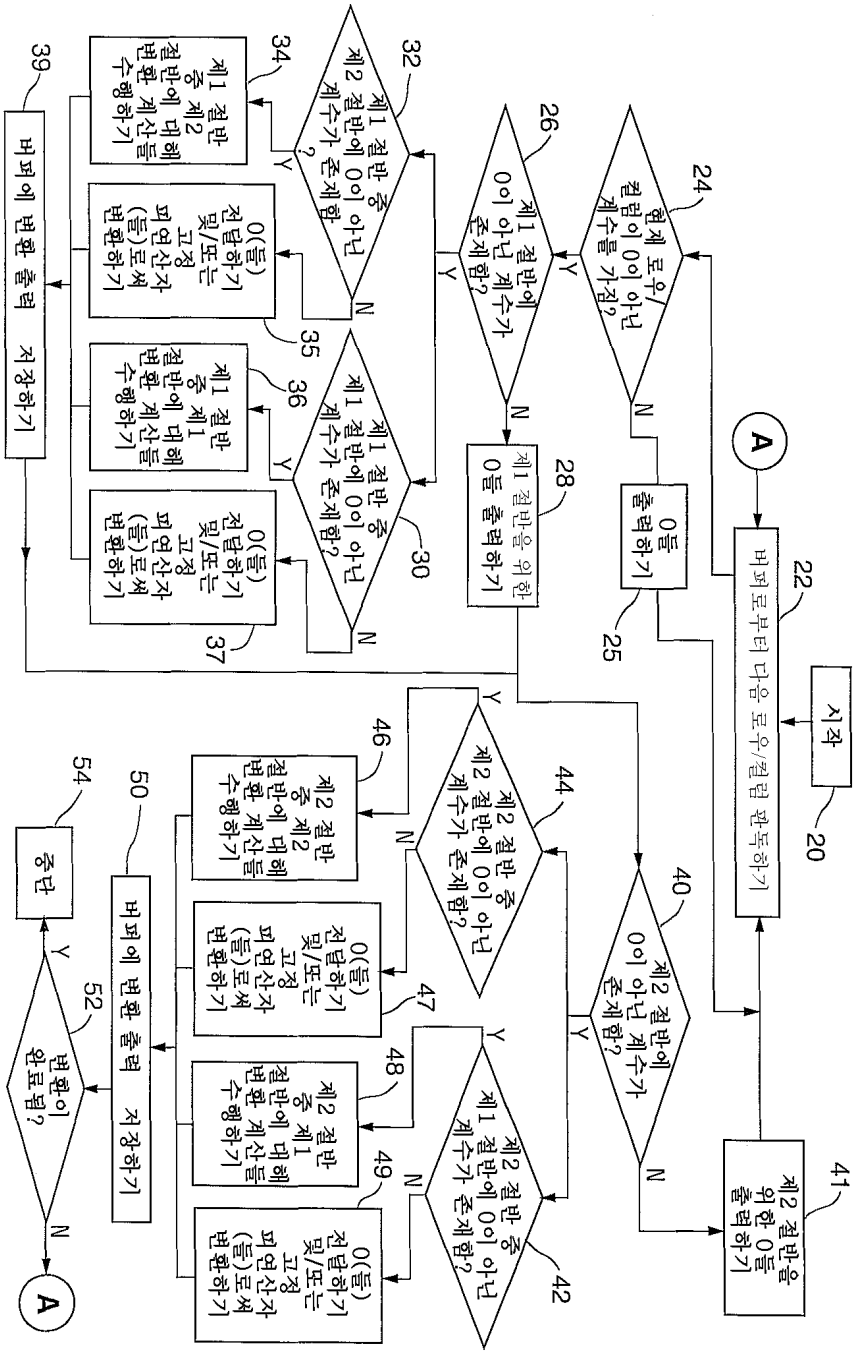
도면2



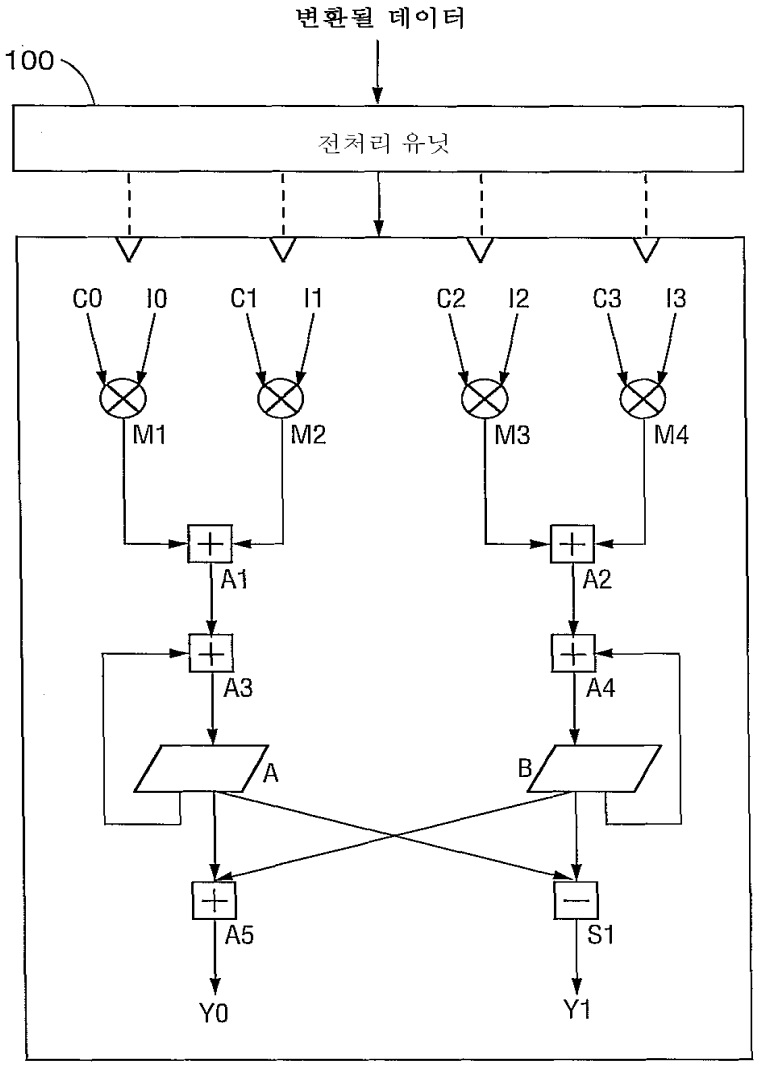
도면3



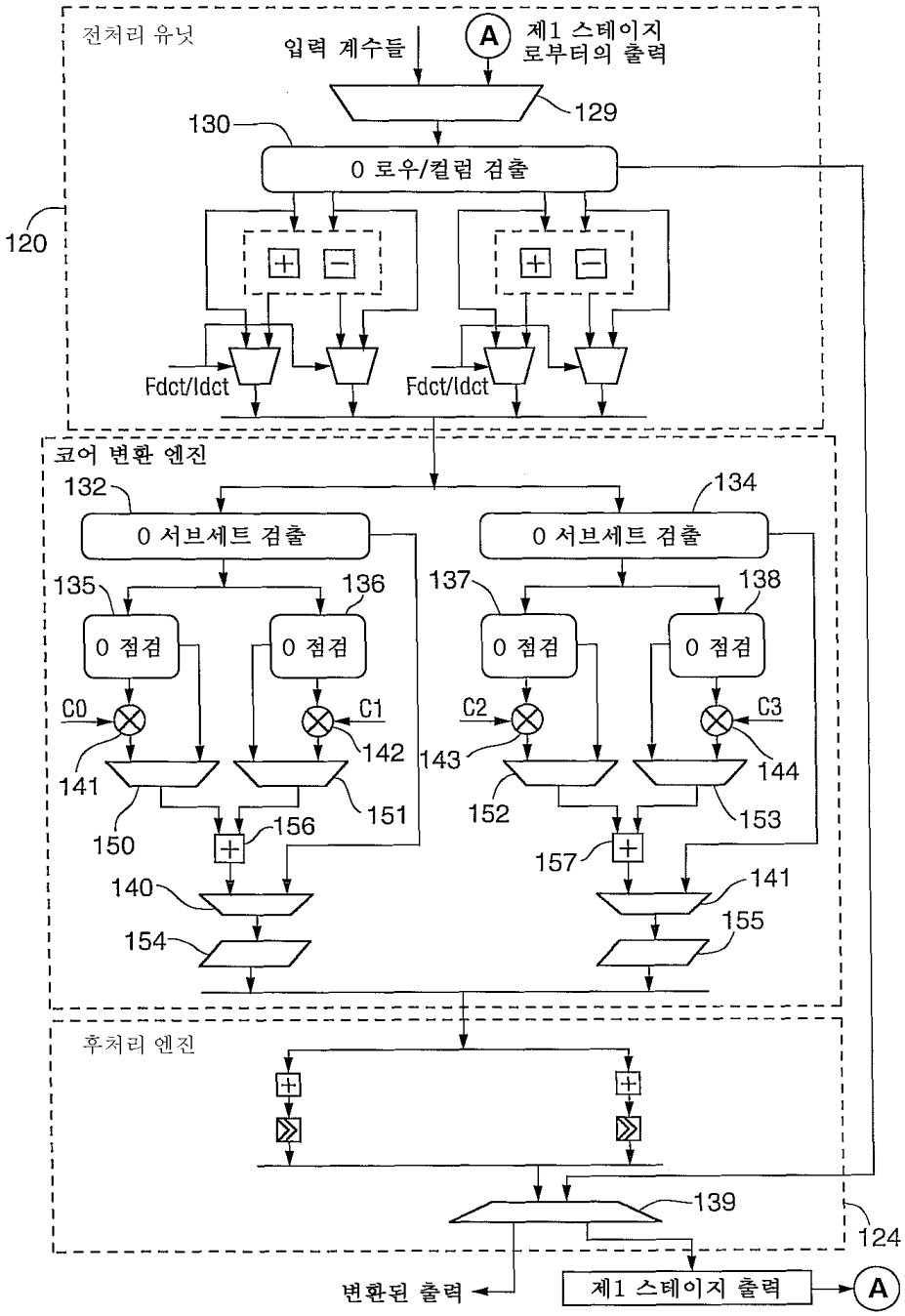
도면4


시작
20
버퍼로부터 다음 로우/컬럼 판독하기
22
A
현재 로우/컬럼이 0이 아닌 계수를 가짐?
24
Y
N
0들 출력하기
25
제1 절반을 위한 0들 출력하기
28
제1 절반에 0이 아닌 계수가 존재함?
26
Y
N
제1 절반 중 제2 계수가 존재함?
32
Y
N
제2 절반에 대해 0(들) 전달하기 및/또는 고정 피연산자 (들)로써 변환하기
35
제1 절반 중 제2 절반에 대해 변환 계산들 수행하기
34
제1 절반 중 제1 계수가 존재함?
30
Y
N
제1 절반에 대해 0(들) 전달하기 및/또는 고정 피연산자 (들)로써 변환하기
37
제1 절반 중 제1 절반에 대해 변환 계산들 수행하기
36
버퍼에 변환 출력 저장하기
39
제2 절반을 위한 0들 출력하기
41
제2 절반에 0이 아닌 계수가 존재함?
40
Y
N
제2 절반 중 제2 계수가 존재함?
44
Y
N
제2 절반에 대해 0(들) 전달하기 및/또는 고정 피연산자 (들)로써 변환하기
47
제2 절반 중 제1 절반에 대해 변환 계산들 수행하기
48
제1 절반에 0이 아닌 계수가 존재함?
42
Y
N
제2 절반에 대해 0(들) 전달하기 및/또는 고정 피연산자 (들)로써 변환하기
49
제2 절반 중 제2 절반에 대해 변환 계산들 수행하기
46
버퍼에 변환 출력 저장하기
50
변환이 완료됨?
52
Y
N
중단
54
A

도면5


변환될 데이터
100
전처리 유닛
C0
I0
C1
I1
C2
I2
C3
I3
M1
M2
M3
M4
A1
A2
A3
A4
A
B
A5
S1
Y0
Y1

도면6


전처리 유닛
입력 계수들
A
제1 스테이지
로부터의 출력
129
130
0 로우/컬럼 검출
120
Fdct/Idct
Fdct/Idct
코어 변환 엔진
132
134
0 서브세트 검출
0 서브세트 검출
135
136
137
138
0 점검
0 점검
0 점검
0 점검
C0
C1
C2
C3
142
143
144
141
151
152
153
150
156
157
140
141
154
155
후처리 엔진
139
124
변환된 출력
제1 스테이지 출력
A