(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[6]
G06F 7/48

(45) 공고일자 1999년04월01일
(11) 등록번호 특0175352
(24) 등록일자 1998년11월10일

(21) 출원번호 특1995-053627
(22) 출원일자 1995년12월21일
(65) 공개번호 특1997-049440
(43) 공개일자 1997년07월29일

(73) 특허권자 한국전자통신연구원 정선종
대전광역시 유성구 가정동 161번지
(72) 발명자 최장식
대전광역시 대덕구 와동 111 현대아파트 103-1404
김기현
대전광역시 유성구 전민동 462-4 나래아파트 103-1402
송경준
대전광역시 유성구 어은동 99 한빛아파트 102-103
김상범
대전광역시 서구 둔산동 1509 크로바아파트 106-1308
(74) 대리인 김영길, 원혜중, 김명섭

심사관 : 민혜정

(54) 8X8 블럭과 2개의 4X8 블럭을 처리하는 이산여현 변환회로

요약

본 발명은 분산산술처리 방식의 정방향 및 역방향 이산여현변환(Discrete Cosine Transform) 회로에 관한 것이다.

일반적인 8x8 DCT는 물론 특정 응용의 하나로 DVCR(Digital Video Cassette Recorder)과 같은 표준을 위한 2개의 4x8 DCT를 모두 지원하는 회로이다.

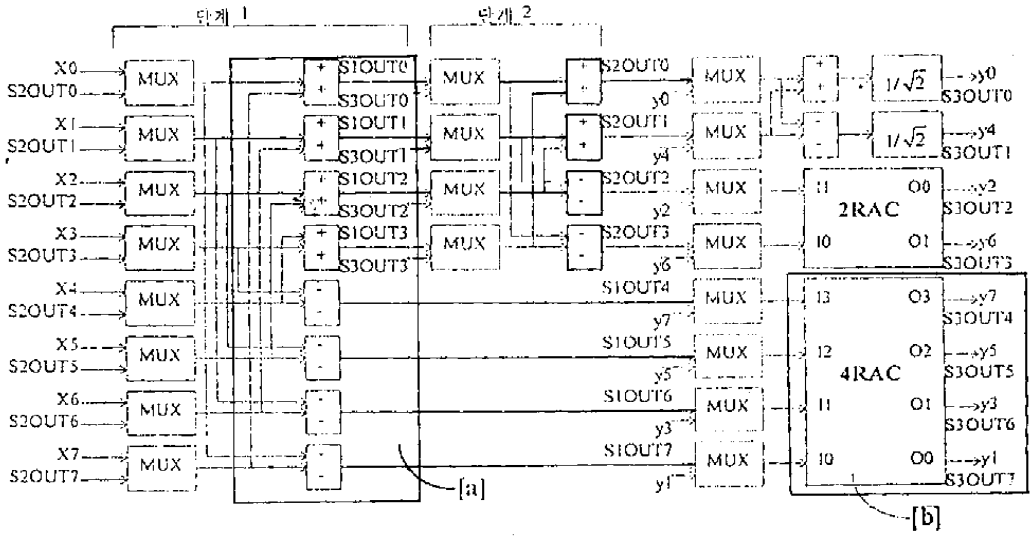
DVCR은 블럭의 움직임 정보에 따라 DCT 블럭의 크기를 달리한다. 움직임이 없는 경우는 8x8 DCT를 수행한다. 반면 움직임이 크다고 판단되면 블럭내 수직방향의 이웃하는 두 화소끼리 합과 차를 계산하여 2개의 4x8 DCT를 수행한다. 8x8 DCT, 2개의 4x8 DCT를 모두 지원하기 위해서는 기존 8x8 DCT 회로에다 2개의 4x8 DCT를 위한 회로가 추가되야 한다.

추가되는 회로로는 수직방향의 이웃하는 두 화소끼리 합과 차를 계산하기 위한 셔플회로와 분산산술방식의 RAC 내에 2개의 4x8 DCT를 위한 ROM이 제공되야 한다.

그리고 8x8 DCT와 2개의 4x8 DCT를 상황에 맞게 선택할 수 있도록 멀티플렉서 회로가 삽입되어야 한다.

본 발명은 이와 같은 변형된 구조를 채택하여 정방향 및 역방향 DCT를 모두 만족시킨다.

대표도



명세서

[발명의 명칭]

8x8 블럭과 2개의 4x8 블럭을 처리하는 이산여현 변환회로

[도면의 간단한 설명]

제1도는 종래의 8x1 DCT 구조도로 분산산술 처리방식의 정방향 역방향 이산 여현 변환회로.

제2도는 종래의 8x1 DCT를 이용한 8x8 DCT로 동작시키기 위한 회로로 8x8 DCT를 처리하기 위해 8x8 블럭에 대해 먼저 행순으로 8x1 DCT를 수행하고 행열 치환한 다음 열순으로 8x1 DCT를 수행하는 회로.

제3도는 크기가 서로 다른 8x8 블럭과 2개의 4x8 블럭을 처리 하기 위해 제1도의 (A)를 변경한 구조.

제4도는 크기가 서로 다른 블럭 처리를 위한 제1도의 (B)를 변경한 구조.

제5도는 크기가 서로 다른 블럭 처리를 위한 2-D DCT 구조.

[발명의 상세한 설명]

본 발명은 분산산술처리 방식의 정방향 및 역방향 이산여현변환(Discrete Cosine Transform)외로에 관한 것으로, 일반적인 8×8 DCT를 처리하는 회로에 2개의 4×8 DCT를 처리하는데 필요한 회로를 추가하여 입력 블럭의 크기에 따른 DCT의 동작 모드를 제어하여 8×8 블럭과 2개의 4×8 블럭을 처리하는 이산여현 변환회로에 관한 것이다.

일반적으로, 기존의 8×1 DCT관련 기술은 미국 US5357-453, 1994.10.18(특허 등록)과, 한국 26634, 1992.12.30(특허 출원)은 분산산술처리(Distributed Arithmetic)방식의 정방향 및 역방향 이산여현변환(Discrete Cosine Transform: 이하 DCT라함)회로에 나타나 있으며, 상기 특허들은 실시간에 영상신호를 주파수신호로 변환할 수 있는 DCT 회로를 효율적으로 구현할 수 있는 방법을 제시하고 있다.

분산산술처리(Distributed Arithmetic) 방식을 이용한 1차원 DCT 구조는 첨부한 제1도에 도시되어 있는 바와같이 구성되며, 상기 종래기술 자료에는 8×1 DCT의 정방향 및 역과정의 동작 과정과 내부 구성 모듈에 대해 자세히 설명하고 있다.

상기 제1도에 도시되어 있는 바와같은 기존 특허의 8×1 DCT 구조는 8×8 DCT로 확장할 수 있는데, 그 구성예가 첨부한 제2도에 도시되어 있다.

통상적으로, 8×8 DCT이 구현은 8×8 입력 블럭에 대해 먼저 행순으로 8×1 DCT를 행하고, 다시 그 값을 열순으로 치환한 후 8×1 DCT를 하는 방법이 대표적이라 할 수 있다.

물론, 행순으로 DCT 변환한 다음 열순으로 DCT 변환하기에 앞서 행렬 변환시키는 회로가 삽입된다.

따라서, 8×8 DCT는 기존 특허인 8×1 DCT가 2개 사용되고 행과 열의 치환을 담당하는 회로의 구현이 상기 제2도와 같이 구성된다.

역방향의 동작은 정방향의 역순으로 열순의 8×1 IDCT가 DCT 변환계수에 대해 이루어지고, 행렬 치환된 후 8×1 IDCT가 수행되어 원래의 픽셀들을 복원한다.

기존 특허의 8×1 DCT가 단일 구조로 정방향과 역방향을 모두 지원하기 때문에 8×8 DCT에 있어서 별도의 회로 추가는 필요하지 않다.

지금까지는 기존 특허의 8×1 DCT 구조를 이용하여 2차원 8×8 DCT의 동작하는 과정에 대해 설명하였다.

이미지 압축 코딩에서 DCT가 처리하는 블럭의 크기에 따라 압축의 효율이 달라진다.

블럭의 크기가 적으면 쉽게 처리되나 압축의 효율은 저하된다.

반대로 블럭의 크기가 크면 압축 효율은 증대된다.

압축 효율 측면에서 이미지의 크기를 크게 하는 것이 좋으나 취급하는데 다소 어려운 점이 있다.

DCT가 처리할 수 있는 블럭의 크기는 2x2, 4x4, 8x8, 16x16, 32x32가 있다.

이미지 압축/복원 코딩의 표준 방식에서는 앞서 열거한 블럭 크기 중에서도 4x4와 8x8이 널리 사용되고 있다.

특히, JPEG, H.261, MPEG 표준에서는 DCT가 8x8 블럭만을 처리하도록 규정하고 있다.

그러나, DVCR(Digitel Video Cassette Recorder)과 같은 규격에서는 블럭의 움직임에 따라 DCT의 입력 블럭의 크기가 달라진다.

움직임이 없는 경우는 8x8 DCT를 취하는 반면에 움직임이 크다고 판단되면 블럭내 수직 방향의 이웃하는 두 화소끼리 합과 차를 계산하여 2개의 4x8 DCT를 하도록 하고 있다.이처럼 서로 다른 블럭 크기를 처리하도록 하는 이유는 코딩 효율과 화질을 향상시키기 위한것인데, 통상의 8x8 DCT를 처리하는 회로에서 2개의 4x8 DCT를 처리하기 위해서는 상당 부분의 추가 구성이 필요하다.

본 발명의 목적은 기존 특허인 8x8 블럭을 처리할 수 있는 구조에다 코딩 효율과 화질을 고려하여 블록의 크기에 따라 8x8 블럭과 2개의 4x8블럭을 함께 효율적으로 처리하기 위한 이산여현 변환회로를 제공하는데 있다.

상기 목적을 달성하기 위한 본 발명의 특징은, 1차원 DCT의 셔플 회로에서 가/감산기 입력단에 입력되는 값들을 블럭 크기에 따라 선택하게끔 하는 멀티플렉서 회로와, 4 RAC 구조에 블럭 크기에 따른 매트릭스 곱셈을 하기 위해 각 블럭용 코사인 매트릭스를 ROM의 형태로 모두 지원하면서, ROM의 출력값을 선택적으로 사용 가능하게 하는 멀티플렉서 회로가 추가되어 정방향 및 역방향 DCT를 처리함으로써, 서로 다른 두개의 입력 블럭

을 처리하기 위해 서로 차이가 생기는 부분과 공통 부분이 한 호로 상에서 모두 지원되고 제어를 위한 8x8 블럭과 2개의 4x8 블럭을 처리하는 이산여현 변환회로를 제공하는데 있다.

이하, 첨부한 도면을 참조하여 본 발명에 따른 바람직한 실시예를 설명한다.

우선, DVCR 규격의 DCT 수식에 대해 살펴보면 다음과 같다.

2개의 4x8 DCT의 정방향 정의식은 아래의 식(1)과 같이 정의된다.

$$F(u, v) = \frac{1}{4} C(u)C(v) \sum_{y=0}^{3} \sum_{x=0}^{7} [f(x, 2y) + f(x, 2y+1)] C_f(x, u) C_f(y, v)$$

$$F(u, v+4) = \frac{1}{4} C(u)C(v) \sum_{y=0}^{3} \sum_{x=0}^{7} [f(x, 2y) - f(x, 2y+1)] C_f(x, u) C_f(y, v)$$

..... 식(1)

또한, 2개의 4x8 DCT의 역방향 정의식은 아래의 식(2)와 같이 정의된다.

$$F(x, 2y) = \frac{1}{4} \sum_{y=0}^{3} \sum_{x=0}^{7} C(u)C(v)[f(u, v) + f(u, v+4)] C_f(x, u) C_f(y, v)$$

$$F(x, 2y+1) = \frac{1}{4} \sum_{y=0}^{3} \sum_{x=0}^{7} C(u)C(v)[f(u, v) - f(u, v+4)] C_f(x, u) C_f(y, v)$$

$$C(i) = \begin{cases} \frac{1}{2\sqrt{2}}, & \text{if } i=0 \\ \frac{1}{2}, & \text{if } i \neq 0 \end{cases}$$

..... 식(2)

상기 식(1)과 식(2)에서 $C_f(x,u)$와 $C_f(y,v)$는 각각 아래의 식(3)과 같이 정의된다.

$$C_f(x, u) = \cos \frac{(2x+1)\pi u}{16}$$

$$C_f(y, v) = \cos \frac{(2y+1)\pi v}{8} \quad \text{..... 식(3)}$$

정방향의 상기 식(1)에서 첫번째 8x1 DCT는 수직 방향의 합과 차가 열순으로 코사인 매스릭스와 곱해진다.

이 결과는 행렬 치환된 후, 2개의 4x1 DCT에 입력되어 최종 DCT 변환된 결과치를 출력한다.

역방향의 식(2)에서 2개의 4x1 DCT가 수행되고 행렬 치환된 후 8x1 IDCT를 거친 후 수직 방향의 합과 차가 계산된다.

이런 처리과정을 가지는 회로의 직접 구현은 기존의 8x1 DCT 구조를 이용한 8x8 DCT의 구조에서 첫번째 I-D DCT에 수직 방향의 합과 차를 구하기 위한 회로가 추가되어야 하고, 두번째 I-D DCT의 셔플회로에서 제1도의 단계1의 가감산 회로가 필요하지 않게 된다.

이 처리 과정은 기존 특허의 구조를 변경하지 않고 직접 이용할려고 하는 의도와 정면으로 배치되고, 정방향과 역방향 동작에 따른 구조가 달라 단일 구조를 가지지 못한다.

또한, 본 발명에서는 기존의 8x1 DCT 구조를 크게 변경하지 않고 8x8 DCT와 2개의 4x8 DCT로 동작 가능한 구조를 제시하는데 역점을 두고 있다.

2개의 4x8 DCT의 동작에 대해 구체적으로 살펴보기로 한다.

상기 식(1)의 정방향 처리시 수직 방향의 합과 차가 코사인 매트릭스와 곱해지는 대신 [f(x,2y)±f(x,2y+1)]는 f(x,2y)와 f(x,2y+1)로 분리된다.

분리된 [f(0,2y)...f(7,2y)]와 [f(0,sy)...f(7,2y+1)]의 8x1 입력 벡터는 $C_f(x,u)$의 8x8 매트릭스와의 곱셈을 통해 8x1 DCT의 결과치를 출력한다.

이때, 8x1 DCT는 각 열에 대해 수행되어 8x8 매트릭스를 생성한다.

상기 매트릭스의 각 행에 대해 두번째 DCT에서 [f(0,2y)±f(0,2y+1)]의 가감산이 이루어진다.

즉, 이전 8x1 DCT에서 수행되어져야 하는 합과 차를 구하는 계산이 현 단계에서 수행되게 한 것이다.

물론 이는 기존의 8x1 DCT 구조를 크게 변경하지 않게 하기 위한 것이다.

이 가감산의 결과치는 각각 2개의 4x1 입력 벡터가 되어 4x4 매트릭스인 와 곱해져 2개의 4x1 DCT 결과치가 생성된다.

상기 식(2)의 역방향 동작 과정은 다음과 같다.

역방향 DCT를 위한 8x1 입력 벡터가 2개의 4x1 벡터로 나누어진 후 4x4 매트릭스와 곱해진다.

이 결과는 다시 합과 차를 구하는 과정을 거친 후 행렬 치환된다.

이 치환된 값은 8x1 DCT를 거쳐 2개의 4x8 역방향 DCT의 결과치를 출력한다.

이 과정은 정방향의 역과정으로 정방향시 사용한 회로를 그대로 사용하며, 단지 데이타의 전달경로만 정방향의 역으로 변경해주면 된다.

8×8 DCT와, 2개의 4×8 DCT를 처리하기 위한 두번째 I-D DCT 구조는 8×8 DCT는 동일한 8x1 DCT 2개를 사용하여 구성된다.

8x8 DCT는 8x1 DCT 2개와 행렬치환을 통해 입력 픽셀 블럭과 DCT 변환 계수간의 정방향/역방향 변환을 수행한다.

이때 2개의 8x1 DCT는 기존의 8x1 DCT를 사용한 것과 거의 동일한 구조로, 단지 입출력 연결(interface) 부분에서만 다르다.

그러나, 2개의 4x8 DCT에서는 서로 다른 2개의 8x1 DCT로 구성된다. 첫 8x1 DCT 구조는 기존의 8x1 DCT 구조와 같으나, 두번째 8x1 DCT의 구조는 2개의 4x1 DCT를 처리하기 위한 것으로 기존의 8x1 DCT 구조와 다르다.

그러면 이와 같이 블럭의 크기에 따라 서로 다른 구조를 가지는 2개의 2-D DCT를 한 회로 상에서 구현하고자 할 경우 첫 8x1 DCT는 같은 구조를 가지므로 기존의 8x1 DCT를 직접 사용하면 된다.

그러나, 두번째 8x1 DCT에서는 블럭 크기에 따라 다른 구조를 가지므로 블럭 크기에 따른 구조상의 차이를 모두 지원하는 회로가 사용되어야 한다. 두번째 8x1 DCT의 구조 변경은 크게 매트릭스 곱셈을 위한 전처리(정방향)/후처리(역방향(단계의 회로와, 셔플회로에서 생성된 벡터와 코사인 매트릭스의 곱셈을 하는 과정에서 코사인 매트릭스의 내용값의 차이에 따른 분산산술 처리의 RAC 회로에서 이루어진다.

먼저 전처리/후처리 단계의 셔플 회로의 차이점에 대해 살펴본다.

8x8 DCT에서 전처리/후처리 단계의 계산은 단계1에서 아래의 식(4)와 같이 연산되는데,

$$A(1)=X0+X7,\ A(2)=X1+X6,\ A(3)=X2+X5,\ A(4)=X3+X4$$
$$A(5)=X3-X4,\ A(6)=X2-X5,\ A(7)=X1-X6,\ A(8)=X0-X7$$

..... 식(4)

의 수식이 계산되고 이 값은 단계2에 입력되어 아래의 식(5)와 같이 계산하게 된다.

$$B(1)=A(1)+A(2),\ B(2)=A(3)+A(4)$$
$$B(3)=A(3)-A(4),\ B(4)=A(1)-A(2)$$ ..... 식(5)

그러나 2개의 4x8 DCT의 전처리/후처리 단계의 계산에서는 첫번 1-D DCT에서 계산되어야 하는 수직방향의 합과 차를 구하는 계산이 두번째 1-D DCT에 포함되므로 단계1의 A(1)에서 A(8)을 구하는 계산식이 아래에 표현한 식(6)과 같이 변경된다.

$$A(1)=X0+X1,\ A(2)=X2+X3,\ A(3)=X4+X5,\ A(4)=X6+X7$$
$$A(5)=X0-X1,\ A(6)=X2-X3,\ A(7)=X4-X5,\ A(8)=X6-X7$$

..... 식(6)

단계2부터는 8x8 DCT에서와 같이 진행된다.

이처럼 A(1)에서 A(8)의 서로 다른 계산을 모두 지원하기 위해서는 전처리/후처리 단계의 단계1의 가/감산기 입력단에 입력을 모드에 따라 선택적으로 공급하기 위한 멀티플렉서 회로가 추가되어야 한다.

이 회로는 첨부한 제3도와 같이 구성되며 2비트 멀티플렉서가 12개 사용된다. 상기 제3도에 도시된 회로를 수식으로 표현하면 아래의 식(7)과 같이 표현된다.

$$
\begin{aligned}
M(1H) &= X0A + X0\overline{A}, & M(1L) &= X7A + X1\overline{A} \\
M(2H) &= X1A + X2\overline{A}, & M(2L) &= X6A + X3\overline{A} \\
M(3H) &= X2A + X4\overline{A}, & M(3L) &= X5A + X5\overline{A} \\
M(4H) &= X3A + X6\overline{A}, & M(4L) &= X4A + X7\overline{A} \\
M(5H) &= X3A + X0\overline{A}, & M(5L) &= X4A + X1\overline{A} \\
M(6H) &= X2A + X2\overline{A}, & M(6L) &= X5A + X3\overline{A} \\
M(7H) &= X1A + X4\overline{A}, & M(7L) &= X6A + X5\overline{A} \\
M(8H) &= X0A + X6\overline{A}, & M(8L) &= X7A + X7\overline{A}
\end{aligned}
$$

..... 식(7)

상기 식(7)에서 M(1H),M(1L)....M(8H),M(8L)의 계산 결과는 전처리/후처리 단계의 첫번째 가/감산기의 입력단에 연결된다.

H는 가/감산기의 상위 입력단에, L은 하위 입력단에 연결된다.

A는 8x8 DCT 모드를, $\overline{A}$ 는 2개의 4x8 DCT 모드를 의미한다.

위의 수식에서 같은 입력을 가지는 M(1H), M(3L), M(6H), M(8L)의 경우는 멀티플렉서 회로를 사용하지 않고 직접 연결된다.

또한, 2개의 서로 다른 입력 블럭 크기를 처리하기 위해서 매트릭스 곱셈시 입력 벡터와 곱해지는 코사인 매트릭스는 입력 블럭 크기에 따라 달리 사용된다.

8x8 DCT에서 2개의 8x1 DCT에 사용된 코사인 매트릭스는 같다.

반면 2개의 4x8 DCT의 경우 첫번째 8x1 DCT에서의 매트릭스 곱셈은 8x8 DCT에서 사용된 코사인 매트릭스를 가지고 이루어진다.

그러나, 2개의 4x1 DCT에서는 8x1 DCT의 상위 계산에서 사용된 4x4 코사인 매트릭스가 각각 사용되어 하위 4x1 DCT에서의 매트릭스가 기존의 8x1 DCT에서 사용된 4x4 코사인 매트릭스와 다르다.

기존 8x1 DCT에서의 하위 코사인 매트릭스 값과 하위 4x DCT에서의 코사인 매트릭스 값의 차이는 아래의 표와 같다.

[표 1]

8×1 DCT 하위 코사인 매트릭스				하위 4×1 DCT 코사인 매트릭스			
$\cos\theta$	$\cos3\theta$	$\cos5\theta$	$\cos7\theta$	$\cos4\theta$	$\cos4\theta$	$\cos4\theta$	$\cos4\theta$
$\cos3\theta$	$-\cos7\theta$	$-\cos\theta$	$-\cos5\theta$	$\cos2\theta$	$\cos6\theta$	$-\cos6\theta$	$-\cos2\theta$
$\cos5\theta$	$-\cos\theta$	$\cos7\theta$	$\cos3\theta$	$\cos4\theta$	$-\cos4\theta$	$-\cos4\theta$	$\cos4\theta$
$\cos7\theta$	$-\cos5\theta$	$\cos3\theta$	$-\cos\theta$	$\cos6\theta$	$-\cos2\theta$	$\cos2\theta$	$-\cos6\theta$

2개의 블럭 크기를 처리하기 위해서는 제1도의 b인 분산산술처리 방식의 RAC 회로가 변경되어야 한다.

블럭 크기에 따라 ROM으로 구현된 코사인 매트릭스가 모두 하드웨어상으로 지원되어야 하고, 블럭 크기에 따라 멀티플렉서 제어를 통해 해당 코사인 매트릭스를 이용한 매트릭스 곱셈이 이루어지도록 한다.

이와 같은 구조는 첨부한 제4도에 도시되어 있는 바와 같은데, 8x1 DCT를 처리하는 경우보다 2개의 블럭 크기를 처리하기 위해서 4RAC에 사용되 ROM의 수가 2배로 증가하고 16비트 멀티플렉서 회로가 8개 축된다.

또한, DCT가 처리하는 블럭의 크기에 따라 DCT의 동작이 달라진다. 이는 DCT가 사용자가 원하는 블럭을 처리할 수 있음을 의미한다.

이를 위해 하드웨어는 2개의 서로 다른 블럭 크기를 처리하기 위한 회로를 모두 지원해야 한다.

두번째 1-D DCT내에서 전/후 처리기의 셔플회로의 첫번째 가/감산기 앞에서 가/감산기로의 입력을 제어하는 멀티플렉서 회로와, 하위 4x4 매트릭스 곱셈을 하는 부분에서 블럭 크기에 따라 해당 코사인 매트릭스의 값을 선택하기 위한 회로가 필요하다.

이렇게 선택적으로 입력 블럭을 처리하는 2-D DCT의 구조는 첨부한 제5도에 도시되어 있는 바와 같이 구성된다.

상기와 같이 구성 동작하는 본 발명에 따른 8x8 블럭과 2개의 4x8 블럭을 처리하는 이산여현 변환회로를 제공하면, 일반적인 DCT가 대개 8x8 입력 블럭만을 처리하도록 구현이 되고 있기 때문에 만약 다른 입력 크기를 지원하기 위해서는 기존 회로를 크게 변경하거나, 새로 구현해야 하므로 변경되는 부분이 많아지면 하드웨어가 크게 증가하면서 비용이 증가한다는 문제점을 본 발명에서는 약간의 회로 추가를 통해 하드웨어 증가에 따른

부담을 적에 하면서 2개의 서로 다른 입력 블럭을 처리할 수 있는 회로를 제시하고 있는 것이다.

즉, 단지 8개의 ROM 모듈과 데이타 경로 제어를 위한 멀티플렉서의 추가만으로 8x8과 2개의 4x8 블럭을 모두 처리할 수 있게 한다는 효과가 있다.

(57) 청구의 범위

청구항 1

수직으로 서로 이웃하는 화소끼리의 합과 차를 구하는 가/감산기를 다단의 셔플회로로 이루어지고, 분산산술처리에 의한 DCT처리를 위한 2RAC과 4RAC이 포함된 8×1 DCT 회로에, 정/역방향 처리시의 정보전달 경로를 선택하도록 다수의 경로선택용 멀티플렉서를 구비시켜 8×1 DCT/IDCT 회로를 구성하고, 그 8×1 DCT/IDCT 회로 2개 사이에 행렬치환기를 삽입시켜 정방향 및 역방향의 8×8 DCT 처리를 하도록 구성된 이산 여현 변환회로에 있어서, 상기 8×1 DCT/IDCT 회로의 첫번째단 셔플회로의 상기 경로선택수단에서 전달 경로가 선택되어 출력되는 출력중 블럭의 크기에 따라 8×8 블럭은 상기 수직방향으로 서로 이웃하는 두개의 화소값을, 4×8 블럭은 상기 수직방향의 순서에 따른 화소값을 차례로 2개의 호소값을 각각 가/감산기의 입력 단자별로 선택하여 상기 가/감산기에 두 입력씩 입력시키기 위한 다수의 제1멀티플렉서와, 상기 4RAC에서 상기 제1멀티플렉서에서 선택되어지는 블럭의 크기에 다른 매트릭스 곱셈을 각기 달리 하기 위해서, 각 블럭용 코사인 매트릭스를 모두 지원하기 위한 데이타 테이블을 구비하여 각각의 화소별 입력별로 8×1 DCT를 위한 2개의 ROM 및 그 두개의 ROM의 출력을 선택하는 멀티플렉서와; 4×1 DCT를 위한 2개의 ROM 및 그 두개의 ROM의 출력을 선택하기 위한 멀티플렉서; 를 각기 더 포함시켜 구성되어 8×8 블럭과 2개의 4×8블럭을 블럭크기에 따라 선택해서 정/역방향으로 DCT 처리하는 것을 특징으로 하는 8×8 블럭과 2개의 4×8 블럭처리하는 이산 여현 변환회로.

청구항 2

제1항에 있어서, 상기 제1멀티플렉서는, 8×1블럭과 4×1블럭을 선택하기 위한 블럭 선택신호(A,/A)와, 상기 수직방향으로 서로 이웃하는 입력 및 수직방향의 순서에 따른 입력을 각각 앤드조합하는 두개의 앤드 게이트와, 그 앤드 게이트의 출력을 오아 조합하여 상기 가/감산기의 상위입력과 하위입력으로 입력시키는 오아게이트로 이루어져 가/감산기에 입력되는 값들을 블럭의 크기에 따라 선택하도록 구성된 것을 특징으로 하는 8×8 블럭과 2개의 4×8 블럭처리하는 이산 여현 변환회로.

청구항 3

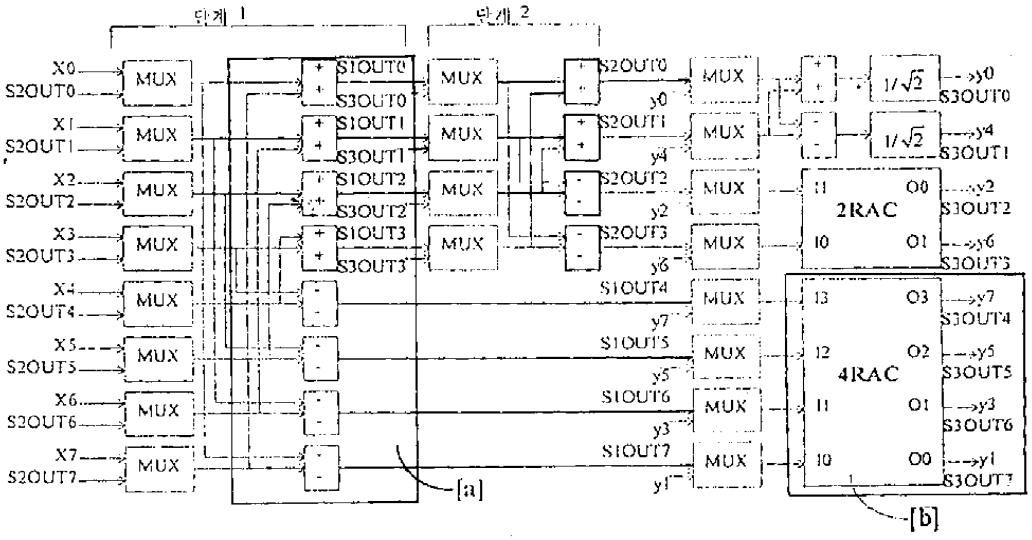
제1항에 있어서, 상기 ROM들은, 8x1 DCT에서의 하위 코사인 매트릭스 값과, 하위 4x1 DCT에서의 코사인 매트릭스 값은,

8×1 DCT 하위 코사인 매트릭스				하위 4×1 DCT 코사인 매트릭스			
$\cos\theta$	$\cos3\theta$	$\cos5\theta$	$\cos7\theta$	$\cos4\theta$	$\cos4\theta$	$\cos4\theta$	$\cos4\theta$
$\cos3\theta$	$-\cos7\theta$	$-\cos\theta$	$-\cos5\theta$	$\cos2\theta$	$\cos6\theta$	$-\cos6\theta$	$-\cos2\theta$
$\cos5\theta$	$-\cos\theta$	$\cos7\theta$	$\cos3\theta$	$\cos4\theta$	$-\cos4\theta$	$-\cos4\theta$	$\cos4\theta$
$\cos7\theta$	$-\cos5\theta$	$\cos3\theta$	$-\cos\theta$	$\cos6\theta$	$-\cos2\theta$	$\cos2\theta$	$-\cos6\theta$

상기 표와 같이 서로 다르게 구성된 것을 특징으로 하는 8×8 블럭과 2개의 4×8 블럭처리하는 이산 여현 변환회로.

도면

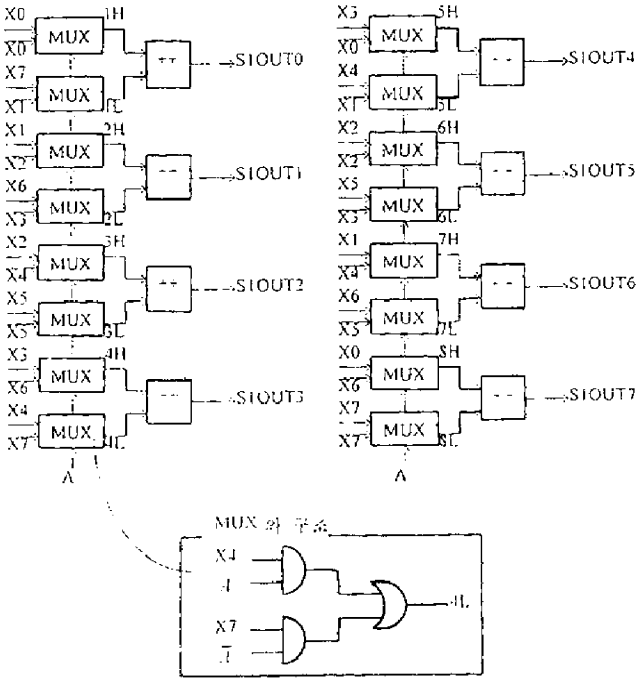
도면1


단계 1
단계 2
MUX
S1OUT0
S1OUT1
S1OUT2
S1OUT3
S1OUT4
S1OUT5
S1OUT6
S1OUT7
S2OUT0
S2OUT1
S2OUT2
S2OUT3
y0
y4
y2
y6
y7
y5
y3
y1
2RAC
4RAC
1/√2
S3OUT0
S3OUT1
S3OUT2
S3OUT3
S3OUT4
S3OUT5
S3OUT6
S3OUT7
[a]
[b]

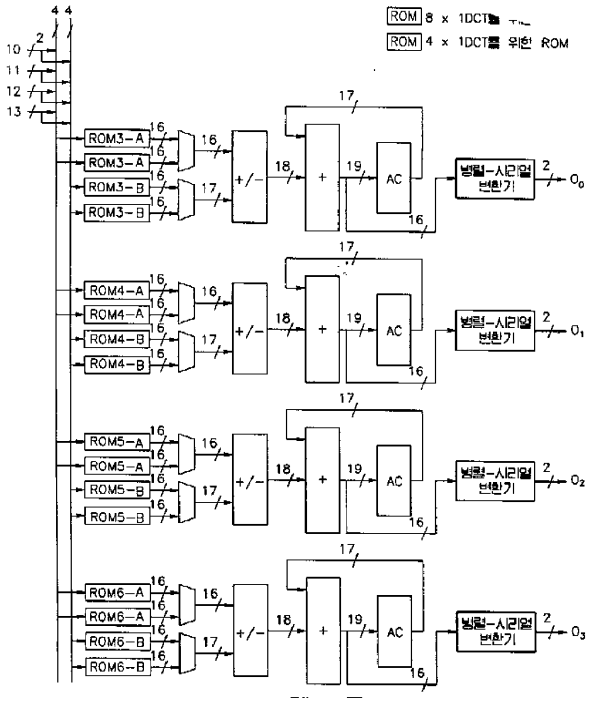
도면2

8x1
DCT/IDCT
행렬치환기
8x1
DCT/IDCT

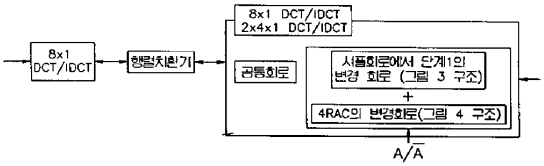
도면3


MUX
S1OUT0
S1OUT1
S1OUT2
S1OUT3
S1OUT4
S1OUT5
S1OUT6
S1OUT7
MUX 의 구조

도면4


ROM 8 x 1DCT
ROM 4 x 1DCT를 위한 ROM
ROM3-A
ROM3-B
ROM4-A
ROM4-B
ROM5-A
ROM5-B
ROM6-A
ROM6-B
+/-
AC
병렬-시리얼 변환기
O_0
O_1
O_2
O_3

도면5


8x1 DCT/IDCT
2x4x1 DCT/IDCT
행렬치환기
공통회로
서플회로에서 단계1의 변경 회로 (그림 3 구조)
+
4RAC의 변경회로(그림 4 구조)
A/A