가산기-기반 분산 연산을 이용한 DCT/IDCT 프로세서 설계

임국찬[O] 장영진 이현수
경희대학교 전자정보학부
{gclim, ddsmurf, hslee}@cann.kyunghee.ac.kr

DCT/IDCT Processor Design using Adder-based Distributed Arithmetic

Guk-Chan Lim[O] Young-Jin Jang Hyon-Soo Lee
Dept. of Electronics & Information, Kyunghee University

요 약

내적을 계산하는데 있어서 Distributed Arithmetic(DA)을 사용하면 곱셈기를 사용하는 것보다 소비전력 및 크기를 효율적으로 줄일 수 있고, 고속동작이 가능한 회로구현이 쉽기 때문에 신호처리 시스템 설계에 많이 사용하고 있다. DA에는 롬-기반 DA와 가산기-기반 DA를 이용한 방법이 있는데, 가산기-기반 DA는 Sharing property와 계수의 Sparse non-zero bit property를 최대한 이용하여 설계가 가능하기 때문에 크기 및 동작속도 측면에서 효율적인 구현이 가능하다.

본 논문에서는 가산기-기반 DA의 이러한 특성을 최대한 이용하여 멀티미디어 신호처리에 적합한 DCT/IDCT 프로세서를 설계하였고 다른 구조 및 롬-기반 DA와 비교 평가해본 결과 크기 및 속도 측면에서 효율적인 결과를 얻었다.

1. 서론

멀티미디어 신호처리의 핵심 기술은 방대한 양의 정보를 효율적으로 저장하고 전송하기 위한 압축 기술이라 할 수 있다. 1974년 DCT 알고리즘[1]이 발표된 이래, 속도 및 크기를 줄이기 위해 많은 알고리즘이 개발되고 VLSI로 구현되었다. 행렬분해를 통한 Chen[2] 알고리즘, 회귀적인 특성을 이용한 Lee[3], Hou[4] 알고리즘 등이 대표적이며, 이론적으로 가장 작은 곱셈횟수를 갖는 Loffler[5] 알고리즘, 시스톨릭 구조를 이용한 알고리즘[6]등이 개발되었다. 이러한 알고리즘들은 다른 형태의 변환 알고리즘에도 적용 가능하다는 장점이 있지만, 곱셈기가 필요하므로 크기와 계산속도 상에 문제점을 가지고있다.

DA는 내적을 구하는데 곱셈기를 사용하지 않고 덧셈기의 수를 줄일 수 있으므로 이러한 문제점을 해결할 수 있다. DA에는 롬-기반 DA와 가산기-기반 DA가 있는데, 이 중 가산기-기반 DA을 이용하면 에러의 허용범위를 고려하여 계수 비트가 '1'인 부분을 최소화하고, 덧셈 항을 최대한 공유하게 함으로서 크기 및 계산속도 측면에서 효율적인 구현이 가능하다[7].

본 논문에서는 가산기-기반 DA 알고리즘을 이용하여 휴대용 멀티미디어 신호처리 시스템에 적합한 DCT/IDCT 프로세서를 설계하였고, Verilog HDL로 시뮬레이션하였다.

2. 기존의 롬-기반 DA를 이용한 DCT/IDCT 구조

DCT는 입력자료의 상관관계를 제거하고 특정한 값에 에너지를 집중시켜 효율적인 압축을 행한다.

8 point 1-D DCT/IDCT 행렬식은 식(1)과 식(2)처럼 각각 2개의 4*4행렬로 나눌 수 있다.

$$\begin{bmatrix} X_0 \\ X_2 \\ X_4 \\ X_6 \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 \\ C_2 & C_6 & -C_6 & -C_2 \\ C_4 & -C_4 & -C_4 & C_4 \\ C_6 & -C_2 & C_2 & -C_6 \end{bmatrix} \begin{bmatrix} x_0+x_7 \\ x_1+x_6 \\ x_2+x_5 \\ x_3+x_4 \end{bmatrix}$$

$$\begin{bmatrix} X_1 \\ X_3 \\ X_5 \\ X_7 \end{bmatrix} = \begin{bmatrix} C_1 & C_3 & C_5 & C_7 \\ C_3 & -C_7 & -C_1 & -C_5 \\ C_5 & -C_1 & C_7 & C_3 \\ C_7 & -C_5 & C_3 & -C_1 \end{bmatrix} \begin{bmatrix} x_0-x_7 \\ x_1-x_6 \\ x_2-x_5 \\ x_3-x_4 \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} = \begin{bmatrix} C_4 & C_2 & C_4 & C_6 \\ C_4 & -C_6 & -C_4 & C_2 \\ C_4 & -C_2 & C_4 & -C_6 \\ C_4 & C_6 & -C_4 & -C_2 \end{bmatrix} \begin{bmatrix} X_0 \\ X_2 \\ X_4 \\ X_6 \end{bmatrix} + \begin{bmatrix} C_1 & C_3 & C_5 & C_7 \\ C_5 & -C_1 & C_7 & C_3 \\ -C_7 & C_5 & -C_3 & C_1 \\ -C_3 & C_7 & C_1 & C_5 \end{bmatrix} \begin{bmatrix} X_1 \\ X_3 \\ X_5 \\ X_7 \end{bmatrix}$$

$$\begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} = \begin{bmatrix} C_4 & C_2 & C_4 & C_6 \\ C_4 & -C_6 & -C_4 & C_2 \\ C_4 & -C_2 & C_4 & -C_6 \\ C_4 & C_6 & -C_4 & -C_2 \end{bmatrix} \begin{bmatrix} X_0 \\ X_2 \\ X_4 \\ X_6 \end{bmatrix} + \begin{bmatrix} C_1 & C_3 & C_5 & C_7 \\ C_5 & -C_1 & C_7 & C_3 \\ -C_7 & C_5 & -C_3 & C_1 \\ -C_3 & C_7 & C_1 & C_5 \end{bmatrix} \begin{bmatrix} X_1 \\ X_3 \\ X_5 \\ X_7 \end{bmatrix} \quad (2)$$

여기서 $C_k = \cos\frac{k\pi}{16}$ 이다.

행렬식에서 알 수 있듯이 DCT/IDCT를 계산하기 위해서는 식(3)과 같이 입력(x_i)와 고정된 계수(C_i)의 곱에 대한 덧셈 즉, 내적을 구해야 한다. 여기서 DA를 이용하면 MAC(Multiply Accumulator)을 사용하는 것보다 효율적인 구현이 가능하다.

$$X=\sum_{i=0}^{N-1} C_i * x_i \tag{3}$$

x_i을 비트 단위로 고려하기 위해 $x_i=\sum_{k=0}^{L-1} x_{i,k}2^{-k}$을 대입하면 아래와 같다.

$$X=\sum_{i=1}^{N-1} C_i*(\sum_{k=0}^{L-1} x_{i,k}2^{-k})$$

$$=\sum_{k=0}^{L-1}(\sum_{i=0}^{N-1} C_i*x_{i,k})2^{-k} \tag{4}$$

$$S=\sum_{i=0}^{N-1} C_i*x_{i,k} \tag{5}$$

식(4)에서 모든 입력 값은 비트 단위($x_{i,k}$)로 연산을 수행하고 연산결과는 Shift-Adder로 구현할 수 있음을 알 수 있다. 전체 블록도는 그림1과 같다.

그림 1. 내적 계산을 위한 전체 블록도

여기서 식(5)의 S를 계산하기 위해 어떻게 DA Unit을 구현하느냐에 따라서 롬-기반 DA와 가산기-기반 DA로 구분할 수 있다.

롬-기반 DA는 입력 비트($x_{i,k}$)들에 대한 모든 경우의 내적 값들을 미리 계산하여 롬에 저장하는 방법이다. 입력 비트들은 롬의 주소로 사용함으로써 계산된 결과를 얻을 수 있다. 그림 2는 N=3일 때, 롬에 저장되는 값들과 전체 구성을 나타낸다.

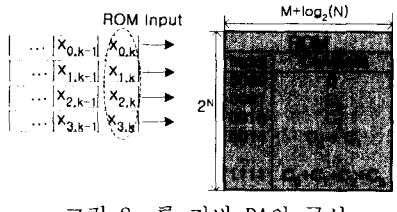

그림 2. 롬-기반 DA의 구성

롬-기반 DA의 가장 큰 단점은 커다란 롬이 필요하다는 것이다. 즉, 그림 2와 같이 N=4일 때 계수들의 합에 대한 경우의 수는 2^N임으로, 이를 저장하기 위한 롬의 크기는 지수적으로 증가함을 알 수 있다. 또한 각 word는 가산된 결과를 정확하게 저장할 수 있도록 충분한 크기를 가져야만 하는데, 일반적으로 $\lceil \log_2(N) \rceil$ 만큼의 추가 비트가 필요하다.

4. 가산기-기반 DA를 이용한 DCT/IDCT 프로세서 제안

본 논문에서는 가산기-기반 DA를 이용하여 8 pior DCT/IDCT 프로세서를 설계하였다. 전체 블록도는 그림에 나타나 있다. 그림 3에서 Butterfly 및 가· 감산기 8*8 계수행렬을 4*4 계수행렬로 분해하기 위해 사용고, 원하는 작업에 따라 데이터의 흐름을 변경할 수 도록 MUX를 사용하여 구성하였다. 여기서 DA Unit A,B각각 4개의 입력에 대한 내적을 구하기 위해 Summation Network로 구성된다.

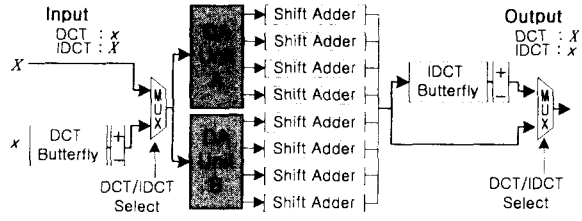

그림 3. 제안한 8 point 1-D DCT/IDCT 전체 블록도

Summation Network을 구성하려면 식(5)의 계수를 비트 단위 고려해야한다. 따라서 식(5)에 $C_i=\sum_{t=0}^{M-1} C_{i,t}2^{-t}$을 대입하면 아래와 같다.

$$S=(\sum_{i=0}^{N-1}\sum_{t=0}^{M-1} C_{i,t}*x_{i,k})2^{-t} \tag{6}$$

식(6)에서 $C_{i,t}$는 고정된 계수이고 $C_{i,t}$=0인 부분은 S 값에 영향을 주지 않으므로, $\sum_{i=0}^{N-1}\sum_{t=0}^{M-1} C_{i,t}$=1인 부분만을 Summation Network로 구성하면 된다.

예를 들어, $C_0=(0110)_2$, $C_1=(0011)_2$, $C_2=(1011)_2$, $C_3=(1000)_2$인 경우를 생각하면 그림 4처럼 $C_{i,t}$=1인 부분의 입력 값이 자릿수에 맞게 더해지도록 구성하면 된다. 이때 그림 4처럼 공통 덧셈 항이 최대가 되도록 묶으면 전체 계산에 필요한 덧셈 항이 줄어들기 때문에 보다 효율적인 구성이 가능하다.

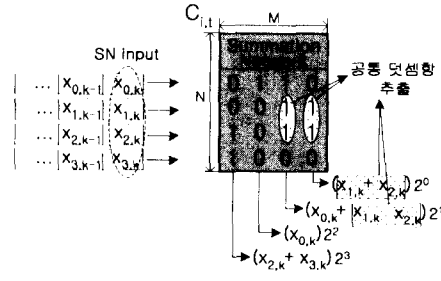

그림 4. $C_{i,t}$의 공통 덧셈부분 추출

그림 5는 3개의 전가산기(FA)와 플립플롭(FF)만으로 구성한 Summation network을 보인다. 롬-기반 DA가 16*6

비트의 롬이 필요하다는 것을 고려하면 크기 및 동작속도 측면에서 훨씬 효율적인 구성이 가능하다.

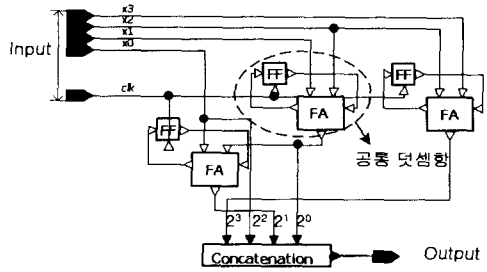

그림 5. Summation Network 구성

5. 시뮬레이션 및 결과 고찰

표 1은 1-D 8 point DCT 알고리즘 계산에 필요한 하드웨어를 비교하였다. 표 1에서 알 수 있듯이 제안된 구조에서는 곱셈기가 필요하지 않으므로 크기 측면에서 매우 효율적임을 알 수 있다.

표 1. 8 point 1-D DCT 계산에 필요한 H/W 비교

	Multiplier	Adder/ Subtracter	Shift Adder	Additonal H/W
[2]	16	26	-	-
[3],[4]	12	29	-	-
[5]	11	29	-	-
[6]	8	16	-	-
ROM-based	-	8	8	ROM
Proposed	-	*8*	*8*	*Summation Network*

표 2. DA Unit-A의 합성 결과 비교

	Number of gate	Data arrive time	Frequency
ROM-based	542	8.29	116.5MHz
Proposed	*390*	*3.96*	*234.9MHz*

또한 롬과 Summation Network의 성능을 비교하기 위한 합성 결과가 표 2에 나타나 있다. 그림 3에서 실제 Shift Adder, 가· 감산기, Butterfly등은 롬-기반 DA와 가산기-기반 DA가 공통적으로 사용하는 부분이므로 평가 부분에서 제외시켰으며 DA unit-A 부분만을 Summation network로 구성하여 롬을 이용하는 경우와 결과를 비교하였다. 입력과 계수는 12bit, 출력은 22bit로 가정하였으며, 2의 보수 표현방식을 사용하였다.

표 2에서 알 수 있듯이 가산기-기반 DA를 이용한 설계가 실제 롬을 사용하는 것보다 크기와 동작속도 측면에서 효율적임을 알 수 있다. 가산기-기반 DA에서는 Summation network의 구성 방법에 따라 성능이 크게 좌우될 수 있으므로, 계수 비트가 Sparse matrix가 되도록 변형하고 덧셈 연산을 최대한 공유하도록 설계하여 효율적인 구조를 갖도록 하였다.

실제 DA Unit-A를 설계할 때, 롬을 사용하는 경우에는 16*14*4 비트의 롬이 필요하지만, 본 논문에서 설계된 Summation network의 경우에는 전가산기와 플립플롭을 각각 16개 사용하였고 200MHz 이상에서 동작이 가능하였다. 따라서 설계된 Summation network는 휴대용 및 실시간 데이터 신호처리 시스템에 적합하다.

그림 7은 그림 3의 DA Unit-A를 합성한 그림이다.

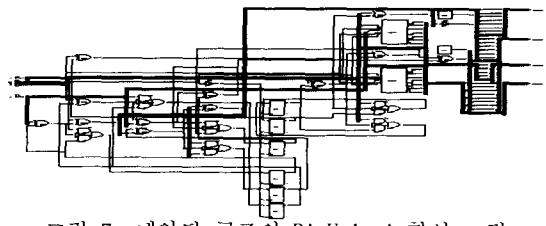

그림 7. 제안된 구조의 DA Unit-A 합성 그림

6. 결론

본 논문에서는 휴대용 멀티미디어 신호처리 시스템에 적합한 DCT/IDCT 프로세서를 설계하였다. 설계된 DCT/IDCT 프로세서는 가산기-기반 DA 알고리즘을 이용하여 구현하였고 다른 알고리즘을 사용하는 것보다 크기 및 동작속도 측면에서 효율적임을 확인하였다.

7. 참고 문헌

[1] N.Ahmed, T.Natarajan and K.R.Rao, "Discrete cosine transform", IEEE Trans. Comput., vol. C-23, pp. 90-23, Jan. 1974

[2] W.H.Chen, C.H.Smith and S.C.Fralick, "A fast computational algorithm for the discrete transform", IEEE Trans. Commun., vol. COM-25, pp. 1004-1008, Sept. 1977

[3] B.G.Lee, "A new algorithm to compute the discrete cosine transform", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, pp.1243-1245, Dec. 1984

[4] H.S.Hou, "A fast recursive algorithm for computing the discrete cosine transform", IEEE Trans. Acoust., Speech, Signal Processing., vol. ASSP-35, pp. 1455-1461, Oct. 1987

[5] Hong Ren Wu and Zhihong Man, "Comments on FAST Algorithms and Implementation of 2-D Discrete Cosine Transform", IEEE Trans. Circuits and System for Video Technology., vol. 8, no 2, April 1998

[6] Sung Bum Pan and Rae-Hong Park, "Unified Systolic Arrays for Computation of the DCT/DST/DHT", IEEE Trans. Circuits and Systems for Video Technology., vol. 7, no 2, April 1997

[7] T.S.Chang, C.Chen and C.-W.Jen, "New distributed arithmetic algorithm and its application to IDCT", IEEE Proc. Circuits Devices System., vol. 146, No.4, August 1999