論文97-34S-1-14

# TMS320C80시스템에서 고속 이산 여현 변환의 해석 및 구현

# (Analysis and Implementation of Fast Discrete Cosine Transform on TMS320C80)

劉 現 範 *, 朴 玄 旭 *

(Hyunbeom Yu and Hyun-Wook Park)

요 약

영상 압축 시스템은 갈수록 그 응용 범위가 넓어지고 있으며, 이에 따라 실시간 처리 시스템에 대한 요구가 갈수록 높아지고 있다. 그러나 실시간 압축/복원은 그 계산량이 크므로 구현상의 문제가 많다. 특히 이산 여현 변환 (Discrete Cosine Transform)과 움직임 측정은 동영상 압축 표준인 MPEG-2의 다른 알고리즘에 비해 그 계산량이 많다. 이 중 기존의 고속 이산 여현 변환 알고리즘들은 덧셈 수와 곱셈 수의 감소에 중점을 두어 왔는데 특히 곱셈이 사이클 수를 많이 소요하고 실제 구현 비용도 많이 든다는 측면에서 주로 감소의 대상이 되어 왔다. 그러나 범용 신호처리 프로세서인 TMS320C80은 특수한 구조를 지니고 있어 알고리즘의 선택이나 구현 측면에서 기존의 방식과는 다른 접근이 필요하다고 할 수 있다. TMS320C80의 기능을 이용하여 알고리즘의 전체 사이클 수를 줄이고 정확도를 높이려면 프로세서의 특성에 적합한 알고리즘을 선택하고 이를 최적화하는 일이 필요하다고 할 수 있다. 이 논문에서는 TMS320C80의 특성에 맞추어 기존의 고속 이산 여현 변환 알고리즘들을 분석하고 그 중 적합한 알고리즘을 선택하였다. 또한 이산 여현 변환 알고리즘이 프로세서의 내부 구조에 최적화되도록 구현하였다. 여러 가지 이산 여현 변환 알고리즘 중 Lee와 Chen의 알고리즘과 같은 1차원 접근 방식이 TMS320C80내부 병렬 처리에 적합한 것으로 판단되었으며, 구현한 결과 이 두 가지 알고리즘 중 Lee의 알고리즘이 속도나 정확도 면에서 더 바람직하다는 결론이 도출되었다. 구현된 고속 이산 여현 변환 알고리즘을 통해 MPEG-2의 실시간 처리 가능성을 검증하고 요구되는 내부 프로세서 ADSP의 갯수를 계산해 보았다.

**Abstract**

There have been many demands for the real-time image compression. The image compression systems have a wide range of applications. However, real-time encoding is hard to implement because it needs a large amount of computations. In particular, the discrete cosine transform (DCT) and motion estimation require a large number of arithmetic operations compared to other algorithms in MPEG-2. The conventional fast DCT algorithms have focused on the reduction of the number of additions and multiplications. The number of multiplications has been especially reduced because it requires more cycles and more expense in realization. Because TMS320C80 has special structure, new approach for implementation of DCT is suggested. The selection of adaptive algorithm and optimization is required to increase system speed and precision using TMS320C80. In this paper, the functions of the TMS320C80 are analyzed and some adaptive DCT algorithms are selected. The DCT algorithms are optimized and implemented. Chens and Lees DCT algorithms among various fast algorithms are selected because 1-D approach is effective in the view of the internal structure of TMS320C80. According to the simulation result, Lees algorithm is more effective in speed and has little difference in precision. On the basis of the result, the possibility of DCT implementation for real-time MPEG-2 system is verified and the required number of the processors, called advanced DSP, is decided for real-time MPEG-2 encoding and decoding.

* 正會員, 韓國科學奇術院 情報 및 通信工學科
(Dept. of Information and Communication Eng. Korea Advanced Institute of Science and Technology)

接受日字: 1996年8月6日, 수정완료일: 1997年1月14日

## Ⅰ. 서 론

현대 사회에 이르러 영상 압축/복원은 영상 전화와 회의 시스템, 원격 의료 진단이나 강의, 각종 감시 시스템 등 그 응용 분야를 넓혀가고 있으며 영상의 크기나 화질에 대한 중요성도 더해가고 있다. 그러나 영상 데이터는 그 양이 매우 많아 실시간 처리에 많은 어려움을 안고 있다. 가장 최근에 표준화된 동영상 압축/복원 표준인 MPEG-2는 HDTV와 같은 고화질 영상을 목표로 하고 있으며 4Mbps이상의 전송부호율(bitrate)를 요구하고 있으나 영상의 크기가 과거에 비해 큰 이유로 실시간 처리에 많은 문제점을 가지고 있다. 이에 따라 고성능 프로세서들의 병렬 처리와 그에 적합한 알고리즘의 구현이 요구된다.[1] 현재 개발 중인 KAIST Image Computing System (KICS)은 실시간 MPEG-2 엔코딩/디코딩을 목표로 하고 있으며 Texas Instrument사의 TMS320C80 DSP 프로세서 5개를 병렬로 이용하는 구조를 가지고 있다.[2],[3]

MPEG-2의 알고리즘은 여러 가지 알고리즘 블럭들의 연속으로 구성되는데 이중 가장 계산량이 많은 알고리즘이 이산 여현 변환과 움직임 측정이다.[4] TMS320C80의 경우 효율적인 연산을 위한 구조를 지니고 있어 적합한 고속 이산 여현 변환 알고리즘을 선택하고 최적화하면 전체 시스템의 속도에 상당한 향상을 기대할 수 있다. TMS320C80의 특성을 간략히 살펴 보면 4개의 내부 프로세서 ADSP는 32비트 연산을 지원하며, 원하는 연산의 비트 수가 적을 때에는 16비트 숫자 2개, 또는 8비트 숫자 4개를 하나의 데이터 레지스터에 입력하여 독립적으로 한 사이클에 연산을 수행할 수 있는데, 이를 split arithmetic operation이라 한다. 또한 한 사이클에 두 곳의 메모리 액세스, 곱셈, 강력한 ALU기능을 동시에 수행할 수 있다. 또한 일반적인 영상 압축 알고리즘을 위한 강력한 DMA기능과 내장 메모리(on-chip memory)가 있으며, 이 기능을 적절히 이용하면 한 8x8블럭의 이미지 데이터를 외부 메모리에서 내장 메모리로 전송해 올 수 있다.

기존의 고속 이산 여현 변환 알고리즘들은 그 방향이 다양하고 주로 곱셈 수와 덧셈 수에 초점을 맞추어 전개되었으며, 곱셈이 덧셈에 비해 사이클 소요가 많고 구현 비용이 크다는 측면에서 주로 감소의 대상이 되어 왔다. 그러나 TMS320C80에서는 덧셈과 곱셈이 한 사이클에 수행되며 또한 동시에 수행되는 강력한 연산 기능을 갖는 특수한 구조를 가지고 있다. 따라서 단순히 곱셈 수나 덧셈 수의 다소만으로 알고리즘의 적합성을 따져서는 안 되고 프로세서에 적합한 알고리즘 구조들을 분석하여 알고리즘을 선택하고 내부 구조에 최적화하여 구현함이 필요하다 하겠다.

이 논문에서는 TMS320C80의 구조를 분석하고 다양한 고속 이산 여현 변환 알고리즘을 연구하여 적합한 알고리즘을 선택하여 구현하였다. 구현 과정에서는 위에서 기술한 특징인 split arithmetic operation, 패킷 전송이라 불리는 영상 처리에 적합한 DMA기능, 내부 모듈-데이터 저장 및 로드, 곱셈, ALU-의 병렬화를 추구하여 최적화하였다. 구현된 알고리즘은 1차원 접근 방식인 Lee와 Chen의 알고리즘이며 이 두 알고리즘 중 Lee의 알고리즘이 속도와 정확도를 종합적으로 살펴볼 때 더 적합한 알고리즘이라고 할 수 있다. Lee의 이산 여현 변환과 역 이산 여현 변환을 구현 하였으며 이때 소요되는 사이클 수를 토대로 MPEG-2 실시간 엔코딩/디코딩 가능성을 검증하였다.

## Ⅱ. TMS320C80의 특성

TMS320C80은 내부 프로세서를 관리하고 부동소수점 연산을 수행하는 1개의 프로세서 (Master Processor)와 계산 전용으로 사용되는 4개의 DSP 프로세서(Advanced DSP:ADSP)를 가지고 있다.[5] 각각의 프로세서는 10Kbyte의 지역 메모리(Local Memory)를 가지고 있으며 다른 프로세서의 지역 메모리도 액세스할 수 있다. 10Kbyte중 2KByte는 명령어 캐쉬(Instruction Cache)이며, 나머지 2KByte는 스택이나 필요한 시스템 정보를 저장하는 데 사용하는 파라미터 메모리로 사용되며, 나머지 6KByte는 사용자가 필요한 입력 데이터와 출력 데이터를 읽거나 저장하는데 사용할 수 있다. 이 때 교차망(Crossbar Network)을 통해 한 사이클에 두 어드레스를 액세스할 수 있으며 32비트, 16비트, 8비트 액세스 모두 가능하다. 외부 메모리에 대한 액세스 속도는 프로세서 속도에 비해 느리므로 실제 데이터의 처리는 내장 메모리의 데이터에 대해서 이루어지는데 이를 위해 다량의 데이터를 한번에 옮겨주는 기능을 하는 Transfer Controller (TC)가 존재한다. TC는 5개의 프로세서로부터 들어오는 패킷 전송 요구들의 우선순위를 정해서 서비스를 수행하는 기능을 한다. TC의 주 기능은 처리

될 데이터를 외부 메모리로부터 내장 메모리로 전송해 주고 처리된 데이터를 다시 외부 메모리로 옮겨주는 것이다. 또한 TC의 기능은 영상 압축 알고리즘들을 위해 8x8블럭과 같이 일정한 간격으로 분리되어 있는 데이터들을 한번에 전송할 수 있는 강력한 DMA기능을 수행한다. 프로세서 내부에서 복잡한 연산이 수행되고 있을 동안 TC는 외부 버스를 통해 내장 메모리의 일정한 영역에 처리할 데이터를 전송해 놓아 처리 시간 이외에 외부 메모리 액세스 시간을 제거하여 전체 처리 속도를 향상시킬 수 있다.

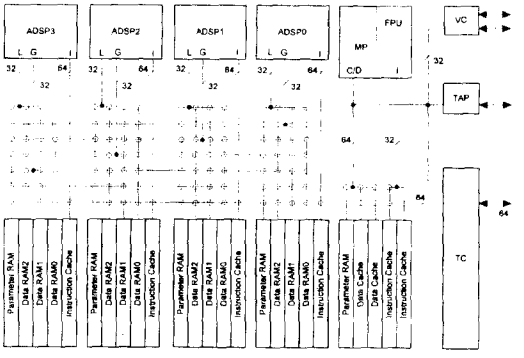



그림 1. TMS320C80의 블럭 다이어그램. 이 프로세서는 네 개의 advanced DSP(ADSP)와 master processor (MP), video controller (VC), transfer controller (TC), test access port (TAP), 메모리, crossbar network으로 구성되어 있다.

Fig. 1. The block diagram of TMS320C80. This chip consists of four advanced DSPs (ADSP), a master processor (MP), video controller (VC), transfer controller (TC), test access port (TAP), and crossbar network connected with internal memory.

각 ADSP는 알고리즘을 구현하는데 중요한 연산 기능과 어드레스 액세스 기능을 가지고 있다. 내부에 곱셈기, ALU, 루프 컨트롤러, 두 개의 어드레스 액세스 기능을 가지고 있으며 각각의 기능은 독립적으로 수행될 수 있다. ALU의 경우 split arithmetic operation과 데이터 쉬프트 기능을 동시에 수행할 수 있으므로 16비트 데이터를 처리하는 이산 여현 변환 알고리즘을 구현하는 데 있어 행(row) 단위나 열(column) 단위로 변환을 수행하는 데 용이하다. 곱셈기의 경우 split arithmetic operation기능을 지원하는 rounded multiplication기능을 가지고 있어 정확도를 향상시키고 연산 비트 수를 정리하여 알고리즘 구현을 용이하게 한다. 즉 위의 모든 기능이 한 사이클에 동시에 수행될 수 있다. 이 점을 활용하면 각 기능을 병렬로 한 사이클에 수행할 수 있는 알고리즘을 선택하고 최적화하여 전체 알고리즘 구현 속도를 향상할 수 있다. 명령어의 길이는 64비트로서 병렬 처리 기능을 지시할 수 있도록 되어 있다. 실에 대부분의 연산을 수행하는 모듈은 ADSP들로서 이 프로세서의 내부 구조를 적절히 이용할 수 있는 일이 알고리즘의 최적화에 중요하다고 할 수 있다.

## Ⅲ. 고속 이산 여현 변환 알고리즘

### 1. 이산 여현 변환의 정의 및 고속 알고리즘의 필요성

영상 압축/복원 표준 안에서 사용되는 8x8블럭 이산 여현 변환은 다음과 같이 정의된다.[16],[17]

$$F(u,v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) \cos[(2x+1)u\pi/16]\cos[(2y+1)v\pi/16]$$

$$f(x,y) = \frac{1}{4} \sum_{x=0}^{7}\sum_{y=0}^{7} C(u)C(v)F(u,v) \cos[(2x+1)u\pi/16]\cos[(2y+1)v\pi/16] \quad \text{(역변환)}$$

여기서

$0 \le u, v, x, y \le 7$,

$x$ , $y$: 이미지 영역에서의 공간 좌표,

$u$, $v$ : 변환 영역에서의 주파수 좌표,

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

위의 수식을 그대로 적용하여 이산 여현 변환을 수행할 경우 한 8x8블럭에 대한 변환을 수행하는 데 1,024번의 곱셈과 896번의 덧셈을 요구로 하며 영상의 크기가 커질 경우 전체 처리 속도에 상당한 부담을 주게 된다. 따라서 이산 여현 변환을 수행하는 데 필요한 사이클 수를 줄이기 위해 FFT ( Fast Fourier Transform)에 의한 방식, 1차원 처리 방식, 행렬 변환을 주로 이용하는 2차원 처리 방식 등 다양한 연구가 진행되어 왔다.

### 2. TMS320C80의 구조에 적합한 고속 알고리즘의 선택

기존의 고속 이산 여현 변환 알고리즘들에 관한 연구는 1차적으로 연산 수의 감소에 중점을 두었으며, 곱

셈이 덧셈에 비해 구현이 복잡하고 가격이 비싸다는 측면에서 곱셈 수의 감소에 중점을 두었다.[8],[9] 그러나 TMS320C80의 경우 고성능 디지털 신호 처리 프로세서(Digital Signal Processor)로서 한 사이클에 덧셈과 곱셈의 동시 수행이 가능하며 이 외에도 , 따라서 기존의 고속 이산 여현 변환 알고리즘들의 곱셈 수와 덧셈 수만을 가지고 비교할 수는 없다. TMS320C80의 경우 대개의 연산이 고정 소수점 프로세서인 ADSP에서 수행된다는 점을 감안하여, ADSP의 특성을 살펴보고 적합한 알고리즘을 선택하였다.

i) 한 사이클에 덧셈과 곱셈이 동시 수행된다는 점에서 곱셈 수와 덧셈 수가 비슷해야 한다는 점이 중요시 되며, 또한 덧셈과 곱셈이 알고리즘 전체에 골고루 섞여 있어야 한다.

ii) ADSP의 데이터 레지스터는 크기가 32비트이며, 이산 여현 변환의 데이터들은 16비트 단위로 처리하면 정확도를 유지하는 데 충분하다. 따라서 하나의 데이터 레지스터에 2개의 16비트 데이터를 로드하여 계산을 수행하면 덧셈 수가 반으로 줄어드는 잇점을 이용할 수 있다.

iii) ADSP는 위에서 언급한 곱셈/덧셈 동시 수행과 더불어 두 개의 데이터를 메모리에 저장 또는 로드할 수 있는 기능을 가지고 있다. 그러나 명령어의 길이가 64비트로 제한되어 있어, 액세스하는 메모리의 오프셋 값이 크면 위의 4가지 기능을 동시에 수행할 수 없다는 단점을 안고 있다. 따라서 가능한 작고 규칙적으로 반복될 수 있는 오프셋 값으로 메모리를 액세스할 수 있는 알고리즘이 적합하다.

iv) 어드레스 오프셋을 작은 값으로 써야 한다는 점이나 Split Arithmetic Operation 측면에서 보았을 때 분리 가능성(Separability)를 이용한 1차원 이산 여현 변환이 적합하다고 할 수 있다. 루프를 이용한 1차원 이산 여현 변환 알고리즘을 구현하면 메모리 액세스할 때 오프셋 값이 작아지고 또 다음 행이나 열에 대해 변환을 수행할 때도 같은 오프셋 값을 사용할 수 있어 효율적이다. 또한 1차원 알고리즘을 선택하면 Split Arithmetic Operation을 이용하여 두 행이나 두 열에 대해 한 번에 변환을 수행할 수 있는 잇점이 있다. 따라서 우선적으로 1차원 이산 여현 변환 알고리즘들이 선택되었고, 이들 중 곱셈과 덧셈의 균형성 측면에서 Chen과 B. G. Lee의 알고리즘을 선택하였다.[10]~[18]

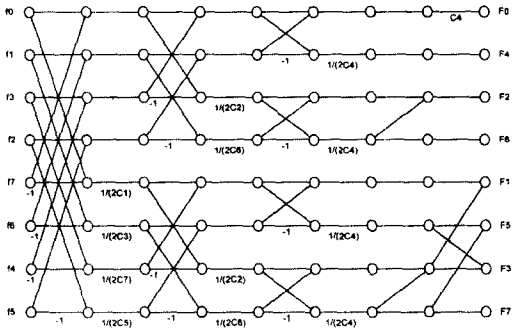



그림 2. Lee의 이산 여현 변환 알고리즘의 흐름도
Fig. 2. The flow graph of Lees DCT algorithm.

## Ⅳ. 구현 방법

### 1. 내장 메모리의 할당과 효율적인 DMA기능의 이용

외부 메모리에 대한 액세스는 알고리즘 속도를 높이는 데 저해 요인이 되므로, 데이터의 처리는 원칙적으로 내장 메모리에 대해서만 수행된다. 내장 메모리는 각 ADSP당 10Kbyte인데 이 중 입력 데이터나 출력 데이터 저장용으로 사용할 수 있는 메모리는 6Kbyte이다. 이산 여현 변환 구현의 경우 2Kbyte의 지역 메모리를 N번째 처리 영역으로 할당하고 또 다른 2Kbyte의 지역 메모리를 (N+1)번째 처리 영역으로 할당하는 것이 효율적이다. 이 때 각각의 메모리의 앞 1Kbyte를 입력 데이터의 영역으로 할당하며 뒷부분의 1Kbyte를 출력 데이터의 영역으로 사용한다. 한번의 패킷 전송으로 옮겨와서 처리하는 데이터는 1Kbyte용량에 해당하는 여덟 개의 8x8블럭이다. TMS320C80의 경우 패킷 전송은 데이타를 3차원으로 읽어서 전송할 수 있는 기능은 가지고 있어 8x8블럭을 외부 메모리에서 내장 메모리로 전송해 오는 문제점을 해결해 준다. 즉, 필요한 데이터가 연속된 어드레스에 존재하지 않더라도 일정한 오프셋을 간격으로 여러 데이터를 전송해 올 수 있다.

앞에서 언급한 바와 같이 TC는 각 프로세서로 부터 패킷 전송 요구를 받아 우선순위를 정하고 이에 따라 데이터를 해당 내장 메모리로 옮겨준다. 이산 여현 변환의 경우 계산 시간이 길어 한 처리 영역에 대해 이산 여현 변환을 수행할 동안 다음 처리 영역에 입력 데이터의 패킷 전송 요구를 해 두면 TC는 해당

ADSP의 처리와 독립적으로 다음 영역에 입력 데이터를 옮겨 놓는다. 따라서 ADSP는 현재 영역에 대한 처리가 끝나자마자 다음 영역의 데이터를 처리할 수 있다.

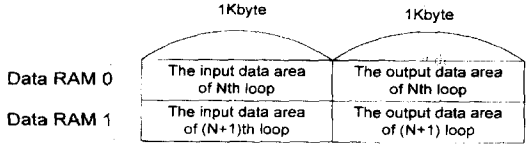



그림 3. 내장 메모리의 처리 영역 할당
Fig. 3. The assignment of processing areas to the on-chip memory.

2. 이산 여현 변환 알고리즘 구현 플로우

이산 여현 변환 알고리즘은 다음과 같이 패킷 전송을 통해 데이터를 내장 메모리에 옮겨 놓고 열 단위로 이산 여현 변환을 수행하고 행 단위로 이산 여현 변환을 수행한 후 결과를 패킷 전송을 통해 외부 메모리에 옮기게 된다. 이때 한번에 처리되는 블럭은 8블럭이다.

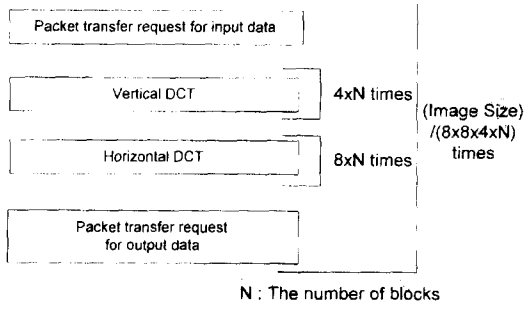



그림 4. 구현된 이산 여현 변환의 루프 구조
Fig. 4. The Loop Structure of the Implemented DCT.

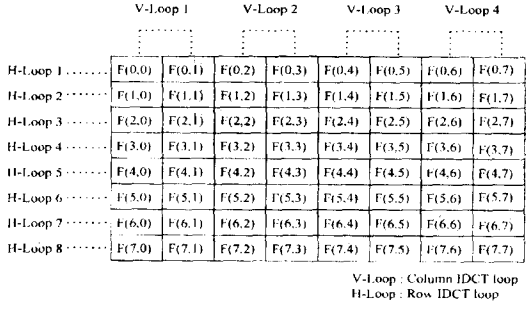



그림 5. 한 8x8 블럭 내에서 split arithmetic operation의 이용
Fig. 5. The use of split arithmetic operation in one 8x8 Block.

3. Split Arithmetic Operation의 이용

열 단위 이산 여현 변환의 경우 데이터가 16비트이므로 두 열의 데이터를 한 데이터 레지스터에 넣고 독립적으로 계산을 수행하며 행 이산 여현 변환의 경우 이웃하는 두 값을 한 데이터 레지스터에 넣고 계산을 수행한다. 이로서 연산의 수를 반으로 줄일 수 있다.

4. Rounded Multiplication의 적절한 사용

ADSP는 Split Arithmetic Operation을 지원하기 위해서 다음 그림과 같은 기능을 갖는 Rounded Multiplication를 지원한다. 데이터 레지스터 안의 두 데이터는 그림과 같이 두 사이클 동안 곱셈을 수행할 수 있으며 이 때 자동적으로 반올림 기능을 수행하여 정밀도를 높일 수 있다. Split Arithmetic Operation과 다른 점은 두 사이클에 두 데이터를 처리할 수 있다는 점이다. 따라서 이 곱셈 기능은 연산 수를 줄이는 데에는 기여하지 못하나 자동 반올림 기능에 의한 정확도 향상과 Split Arithmetic Operation의 중간 과정에서도 16비트 데이터 자동으로 유지해 주는 기능으로서의 역할을 하고 있다.

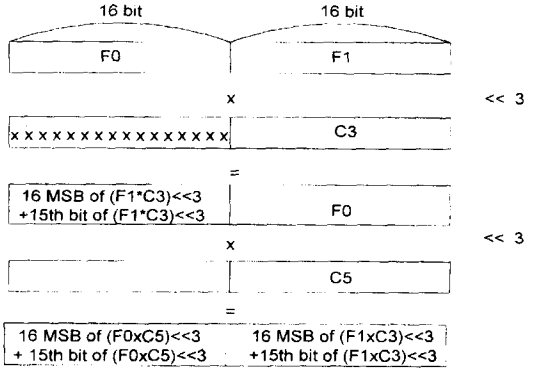



그림 6. Rounded multiplication의 예
Fig. 6. An example of rounded multiplication.

## V. 실험 결과 및 검토

1. 두 알고리즘의 정확도 비교

이산 여현 변환 알고리즘은 TMS320C80의 시뮬레이터를 통하여 구현되었다. 정밀도는 영상 테스트 용으로 사용되는 10가지 범용 영상에 대해서 비교되었다. 부동 소수점 이산 여현 변환과 비교한 결과 PSNR이 51.3~51.6dB이며 ADSP상에서 이산 여현 변환한 데이터를 역 이산 여현 변환하였을 때의 PSNR은 49.0dB

~ 50.0dB이다. Lee의 알고리즘이 Chen의 알고리즘에 비해 PSNR측면에서 약 0.2dB정도 작아 정확도가 약간 낮았으나 모든 영상에 대해 49dB이상으로 화질에는 영향이 없는 정도의 값이라고 할 수 있다. 즉 두 알고리즘 모두 적합하다고 할 수 있다.

2. 두 알고리즘의 속도 측면에서의 비교

실제로 Lee의 이산 여현 변환 알고리즘을 구현하기 전에 TMS320C80에서 두 알고리즘 중 어느 알고리즘이 더 효율적인지 검토하기 위해 Chen과 Lee의 역 이산 여현 변환 알고리즘을 구현하여 비교해 보았다. 이 결과 소요되는 사이클 수에서 Lee의 알고리즘이 더 적합하다는 결론이 도출되었다. Chen의 알고리즘은 덧셈과 곱셈이 골고루 섞여 있지 않아, 한 8x8블럭의 덧셈과 그 다음 8x8블럭의 곱셈을 동시에 수행하는 프로그램 구조를 가지나 구현이 용이하지 않고 사이클 수의 감소에 많은 문제가 있다. 이산 여현 변환의 경우도 Chen의 알고리즘의 구조적인 문제를 고려하여 Lee의 알고리즘이 더 적합하다는 결론하에 Lee의 알고리즘을 구현해보았다. 표1은 Chen과 Lee의 역 이산 여현 변환 알고리즘과 Lee의 이산 여현 변환 알고리즘을 구현하는 데 필요한 사이클 수이다.

표 1. Chen과 Lee의 역 이산 여현 변환 알고리즘을 구현하는 데 필요한 사이클 수

Table 1. The number of cycles of Chens and Lees IDCT.

| | Chen 역이산 여현 변환 | Lee 역이산 여현 변환 | Lee 이산 여현 변환 |
|---|---|---|---|
| Setup | 64 | 58 | 49 |
| Loop Setup | 8 | 8 | 8 |
| Pre-execution of Vertical Loop | 19 | 0 | 0 |
| Vertical Loop | 36x4xN | 38x4xN | 37x4xN |
| Horizontal Loop Setup | 12 | 2 | 2 |
| Horizontal Loop | 56x8xN | 47x8xN | 28x8xN |
| End | 8 | 8 | 8 |

N : 한 번의 패킷 전송에 의해 처리될 8x8블럭의 수

위의 표를 살펴 보면 TMS320C80에 적합한 알고리즘은 내부 병렬 처리에 유리한 구조를 가지고 있는 Lee의 알고리즘임을 알 수 있다. 이산 여현 변환 시에 Split Arithmetic Operation을 사용하지 않는다면, 속도 면이나 정확도 면에서 떨어지는 결과가 나오게 된다. 하나의 16비트 데이터를 한 데이터 레지스터에서 계산할 경우 소요되는 사이클 측면을 살펴 보면, 곱셈 수에 있어서는 변동이 없으나 덧셈의 수는 2배가 된다. 따라서 전체적으로 곱셈의 수에 비해 덧셈의 수가 많아져 내부 병렬 처리에 문제가 있다. 이 결과 덧셈 수가 곱셈 수를 초과하는 만큼 사이클이 더 소요되는 결과가 예측된다. 정확도 측면에서 살펴 보면 한 데이터 레지스터에 두 개의 16비트 데이터가 들어 있을 경우 곱셈 결과를 자동으로 반올림하고 계산 결과를 16비트로 맞추어주는 Rounded Multiplication기능을 적용할 수 있지만 하나의 데이터가 있을 경우 반올림 기능을 수행할 수 없어 정밀도가 낮아지는 결과가 초래된다. 이산 여현 변환의 경우 역 이산 여현 변환과 같이 변환된 값을 경계치 안으로 맞추어주는 과정이 필요하지 않으므로 사이클 수가 훨씬 적게 든다.

3. Lee의 고속 이산 여현 변환 알고리즘을 이용한 실시간 MPEG-2처리 가능성

표2는 TMS320C80의 동작 속도가 40MHz라는 점을 바탕으로 실시간 처리 MEPG-2 시스템에 요구되는 이산 여현 변환의 사이클 수와 ADSP 개수를 계산한 표이다.

표 2. 실시간 MPEG-2시스템에서 이산 여현 변환을 구현하기 위해 소요되는 사이클 수와 ADSP의 개수

Table 2. The cycles and the number of ADSP required for the DCT in the realization of real-time MPEG-2.

| | 이산 여현 변환 | 역 이산 여현 변환 |
|---|---|---|
| 8 8x8블럭 계산에 필요한 사이클 수 | 3,046 | 4,300 |
| MPEG-2규격 한 프레임에 필요한 사이클 수 | 3,084,075 | 4,355,900 |
| MPEG-2규격 한 프레임에 필요한 시간 | 61.68(ms) | 87.12(ms) |
| MPEG-2규격 30 프레임에 필요한 ADSP 수 | 2 | 3 |

위의 결과에서 볼 때 이산 여현 변환의 경우 2개의 ADSP로 실시간 계산이 가능하며, 역 이산 여현 변환의 경우 3개의 ADSP를 이용하면 실시간 계산이 가능하게 된다. MPEG-2 엔코딩 과정에서는 이산 여현 변환과 역 이산 여현 변환을 모두 필요로 하므로 5개의

ADSP가 할당되고 디코딩 과정에서는 역 이산 여현 변환만 수행하면 되므로 3개의 ADSP가 실시간 계산을 수행할 수 있다.

## Ⅳ. 결 론

본 논문에서는 가장 최근의 동영상 압축 표준인 MPEG-2의 모듈 중 고속 이산 여현 변환 알고리즘들을 분석하고 TMS320C80의 특성에 맞추어 선택하고 구현하는 방법을 제시하고 있다. ADSP내부의 병렬 연산 기능을 수행하기 위해서 적합한 알고리즘을 선택하고 곱셈, 덧셈 기능과 메모리 액세스 기능을 병렬화 하였다. 또한 ADSP간의 데이타 전송시간을 줄이기 위해서 패킷 전송을 최적화하여 구현하였다. 이로서 ADSP 내부에서의 각 기능 사이의 병렬 처리와 네 개의 ADSP사이의 병렬 처리를 이용하여 이산 여현 변환을 구현하였다. 고속 알고리즘 중 Lee와 Chen의 알고리즘을 선택하여 구현하고 비교한 결과 정확도와 속도 측면을 종합적으로 고려할 때 Lee의 알고리즘이 Chen의 알고리즘에 비해 TMS320C80의 구조에 더 적합한 것으로 판단되었다. Lee의 알고리즘 구현 결과를 이용하여 MPEG-2실시간 처리를 위한 DCT의 구현이 가능하다는 점이 검증되었으며, 이 때 엔코딩 과정에서는 5개의 ADSP, 디코딩 과정에서는 3개의 ADSP가 필요하다는 결과가 도출되었다.

## 참 고 문 헌


[1] J. M. Jong, H. W. Park, K. S. Eo, M. H. Kim, P. Zhang, Y. Kim, UWGSP4: an imaging and graphicssuper workstation and its medical applications, SPIE Vol. 1653 Image Capture, Formatting, and Display, pp. 422-433, 1992.

[2] T. Akiyama, H. Aono, K. Aoki, K. W. Ler, B. Wilson, T. Araki, et. al., MPEG2 Video Codec using Image Compression DSP, IEEE Trans. on Consumer Electronics, Vol. 40, pp. 466-472, 1994.

[3] K. Shen, L.A. Rowe, and E. J. Delp, A Parallel Implementation of an MPEG1 Encoder: Faster than Real-time, SPIE Proc. Digital Video Compression, 1995.

[4] Generic Coding of Moving Pictures and Associated Audio: Video, ISO/IEC JTC1/SC29/WG11 13818-1,2,3, 1994.

[5] TMS320C80 Users Guide, Texas Instruments, 1994.

[6] A. K. Jain, Fundamentals of Digital Image Processing, pp. 132-154, Prentice-Hall, 1989.

[7] R. C. Gonzales, P. Wintz, Digital Image Processing, pp. 83-144, Addison-Wesley Company, 1992.

[8] K. R. Rao, P. Yip, Discrete Cosine Transform, Algorithms, Advantages, Applications, Academic Press, pp. 48-120, 1990.

[9] W. B. Pennebaker, J. L. Mitchell, JPEG: Still Image 데이터 Compression Standard, Van Nostrand Reinhold, pp. 29-64

[10] W. Chen, C. H. Smith, and S. C. Fralick, A Fast Computational Algorithm for the Discrete Cosine Transform, IEEE Tran. On Commun. COM-25(9):1004-9, Sept. 1977.

[11] B. D. Tseng and W. C. Miller, On Computing the Discrete Cosine Transform. IEEE Trans. Computers. C-27(10):966-8, Oct. 1978.

[12] R. M. Haralick, A Storage Efficient Way to Implement the Discrete Cosine Transform, IEEE Trans. Computers, C-25:764-5, Jul.1976.

[13] S. Winograd, On computing the Discrete Fourier Transform, Mathematics of Computation, 23(141):175-99, Jan. 1978.

[14] B. G. Lee, A New Algorithm to Compute the Discrete Cosine Transform, IEEE Trans. On Acoustics, Speech, and Signal Proc., Vol. ASSP-32, No. 6, Dec. 1984.

[15] Z. Wang, Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform, Aug. 1984.

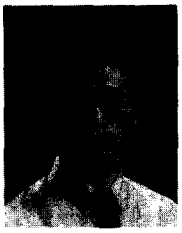
[16] A. Litgtenberg and M. Vetterli. A discrete Fourier-cosine transform chip., IEEE J. On Selected Areas in Commun. SAC-4:49-61, Jan. 1986.

[17] E. Feig, S. Winograd, Fast Algorithms for the Discrete Cosine Transform, IEEE

Trans. on Signal Proc., Vol. 40, No. 9, Sept. 1992.

[18] P. Duhamel and M. Vetterli, Fast Fourier Transforms: A Tutorial Review and a State of the Art, Signal Processing.

## 저 자 소 개



**劉 現 範(正會員)**

1994년 2월 한국과학기술원 과학기술대학 전기및전자공학과 학사. 1994년 9월 ~ 1996년 8월 한국과학기술원 정보및통신공학과 석사. 1996년 8월 ~ 현재 LG전자 멀티미디어연구소 연구원.

**朴 玄 旭(正會員)**

1981년 2월 서울대학교 전기공학과 학사. 1983년 2월 KAIST 전기및 전자공학과 석사. 1988년 2월 KAIST 전기및전자공학과 박사. 1989년 7월 ~ 1992년 3월 Washington Univ. 연구원. 1992년 5월 ~ 1993년 7월 삼성전자 정보컴퓨터연구소 수석연구원. 1993년 8월 ~ 현재 KAIST 정보및통신공학과 부교수. 전공 분야: 디지탈 신호처리, 영상처리, Image computing system, 의료영상