論文93-30B-10-2

재구성 가능한 다중 프로세서 시스템을 이용한 혼합 영상 부호화기 구현에 관한 연구(연구 Ⅱ : 병렬 알고리즘 구현)

(A Study on Hybrid Image Coder Using a Reconfigurable Multiprocessor System(Study Ⅱ : Parallel Algorithm Implementation))

崔 翔 勛*, 李 光 基*, 金 仁*, 李 鎔 均*, 朴 圭 泰*

(Sang Hoon Choi, Kwang Kee Lee, In Kim, Yong Kyun Lee, and Kyu Tae Park)

要 約

연구 I에서 구현한 멀티프로세서 시스템상에 동영상 복합 부호화 알고리즘을 실현하였다. 알고리즘의 가장 효율적인 수행을 위하여 파이프라인기법과 분할기법을 이용하여 복합 부호화 알고리즘을 실현하였고, 각각에 대하여 수행시간, 통신량과 효율을 비교하였다. 구현된 시스템의 성능은 MPEG 부호화 알고리즘을 대상으로 비교, 분석하였다. 이론적 고찰과 실험 결과 분석을 통하여 분할기법은 알고리즘의 수정 및 확장이 용이하고 전체 효율도 파이프라인기법 보다 우수하여, 동영상 부호화에 적합한 병렬처리 구조임을 입증하였다.

Abstract

Motion picture algorithms are realized on the multiprocessor system presented in the Study I. For the most efficient processing of the algorithms, pipelining and geometrical parallel processing methods are employed, and processing time, communication load and efficiency of each algorithm are compared. The performance of the implemented system is compared and analysed with reference to MPEG coding algorithm. Theoretical calculations and experimental results both shows that geometrical partitioning is a more suitable parallel processing algorithm for moving picture coding having the advantage of easy algorithm modification and expansion, and the overall efficiency is higher than pipelining.

Ⅰ. 서론

일반적으로 동영상 부호화를 수행하기 위해서는 많은 양의 연산이 필요하다. 따라서 연구 가에서 제안된 프로세싱 모듈 하나만으로는 동영상 부호화의 고속 수행이 불가능하므로 고속 부호화 시스템을 구현할 경우 병렬처리의 도입은 필요 불가결하다.

병렬처리 구조는 파이프라인기법과 분할기법으로 크게 나누어 생각할 수 있다. 파이프라인기법에서는 처리하고자 하는 데이터가 각기 다른 프로그램이 수행되는 프로세서와 연동되어 전체 알고리즘이 수행되므로, 이 기법을 알고리즘 병렬처리 기법이라고 한다.[1][5][7] 이 기법은 병렬처리 기법중에서 가장 간단한 기법으로써, 이 기법은 해결하고자 하는 문제를

* 正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Yonsei Univ.)
接受日字 : 1993年 3月 23日

시스템에 적절히 분배하는 것이 곤란하고, 모든 문제에 대하여 공통적으로 효율적인 성능향상을 기대하기 어렵다. 또한 특정한 프로세서에서 특정한 데이터와 연산이 수행되므로, 알고리즘의 수정과 확장이 용이하지 못한 구조이다.

데이터 분할기법은 처리하고자 하는 데이터를 시스템내의 모든 프로세서에게 균등하게 분배하여 처리한다. 따라서 이 기법을 지형적(geometric) 병렬처리 기법이라고 한다. 이 기법은 시스템내의 주 프로세서를 제외한 모든 부 프로세서상에서 같은 프로그램이 전체 데이터의 일부분만을 처리하여 결과를 주 프로세서에 전달한다. 이 기법은 알고리즘의 수정과 확장이 용이하고, 프로세서의 갯수를 증가시킴에 따라서 성능이 거의 비례적으로 증가하게 설계할 수 있다. 그러나 각 프로세서가 처리하고자 하는 데이터가 자신의 메모리에 존재하지 않는 경우, 다른 프로세서로부터 데이터를 이동시키는 통신에 대한 부담이 있다. 데이터 분할기법은 프로세서가 처리하고자 하는 데이터가 균등한 경우에는 효율적으로 알고리즘을 수행할 수 있다.

본 연구에서는 파이프라인 기법과 분할기법의 계산량과 통신량을 분석하여, 동영상 부호화시에 분할기법이 유리함을 입증하였고, 동영상 부호화를 분할기법으로 처리하는 경우 입력되는 영상의 통계적 특성이 시간에 따라 불규칙하게 변화하고, 이에 따라서 각 프로세서에서 처리되는 계산량의 차이가 발생하여 각 영역에서의 수행시간은 일정하지 않음으로써 발생되는 부호화기의 전체적인 성능 저하를 분석하였다.

Ⅱ. 동영상 부호화 알고리즘

본 연구에서는 시스템의 성능을 평가하기 위하여 동영상 복합 부호화 기법중에서 MPEG 알고리즘을 채택하였다. MPEG 알고리즘은 다음과 같다.

1. 기본 알고리즘

MPEG 알고리즘은 부호화를 그림 1과 같이 GOP (Group of Picture) 단위로 수행한다. GOP는 I 화면(Intra-coded picture), P 화면(Predictive-coded picture), 과 B 화면(Bidirectionally predictive-coded picture)의 세가지 다른 부호화 형태의 화면(picture)으로 구성되며, 각 화면은 서로 다른 부호화 방식을 취하고, 부호화와 복호화 순서는 화면 표시 순서와 다르다. [2,3]

그림 1의 GOP 구조는 임의로 변경할 수 있다. 즉, I 화면과 I 화면의 간격을 N 이라고 정의하고, I

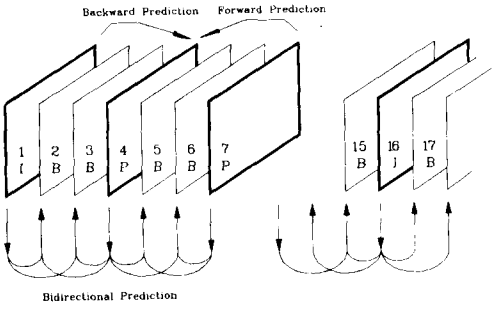


그림 1. GOP 구조
Fig. 1. The GOP structure.

화면과 P 화면 또는 P 화면과 P 화면의 간격을 M 이라고 정의하고, 응용목적과 주어진 비트량에 따라 N 또는 M을 변경할 수 있다. 그림 1은 N=15, M=3 인 경우이다.

MPEG에서는 352×240 크기의 MPEG SIF (Source Input Format) 영상을 slice 단위로 분할하여 부호화하도록 하는데, 각 slice의 크기는 반드시 동일할 필요는 없다. 이 slice는 매크로블럭으로 구성되고, 매크로 블럭은 하나의 16×16 휘도 성분 블럭(네개의 8×8 블럭으로 구성)과 두개의 8×8 색차 성분 블럭으로 구성된다. 매크로 블럭의 휘도 성분은 화면간 부호화시 이동량 검출과 이동 보상의 기본 단위가 된다. 8×8 크기의 블럭은 DCT 및 양자화의 대상이 된다. 위의 계층적 데이타의 구조를 그림 2에 나타내었다.

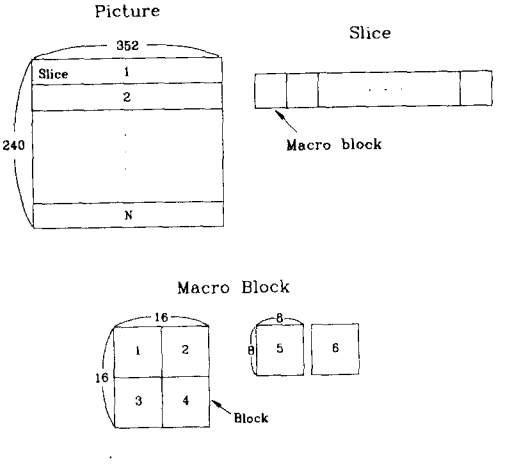


그림 2. 계층적 데이타의 구조
Fig. 2. The hierachical data structure.

부호화기는 부호화할 화면의 GOP내의 위치에 따

라서 매크로 블럭의 부호화 방식을 다르게 할 수 있다. 전체적인 부호화과정을 그림 3에 나타내었다. 복호화는 부호화의 역과정이다.

2. 화면내 부호화(Intraframe coding)

각 GOP의 최초 화면은 화면내 부호화를 수행한다. 이 부호화 방식은 자체 정보만으로 부호화가 가능하며, 복원 후의 전체 화질에 가장 큰 영향을 주며, 비순차 접근시 기준 화면으로 사용되고 가장 많은 비트를 생성한다.

MPEG의 화면내 부호화는 정지영상 부호화 국제 표준안인 JPEG 부호화 방식과 유사하고, 부호화 과정은 입력된 영상에 대하여 DCT를 수행하고, DCT 계수를 양자화한 후에, zigzag scanning과 Huffman 부호화를 수행한다. JPEG 알고리즘과의 차이는 화질과 버퍼의 상태를 고려하여, 각 매크로블럭 마다 다른 양자화 단계수를 부여할 수 있다.

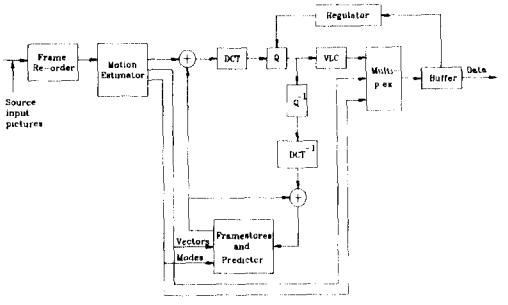


그림 3. MPEG 부호화기 구성도
Fig. 3. Block diagram of the MPEG encoder.

그러나, 화면내 부호화 방식만으로 약 100:1 정도의 큰 압축율과 동시에 좋은 화질을 유지하기가 어렵다. 따라서, 높은 압축율을 얻기 위해서 MPEG 알고리즘은 영상회의와 영상전화를 위한 부호화 알고리즘인 CCITT H.261 부호화 방식과 유사한 화면간 부호화 방식을 사용한다. CCITT H.261과 다른 점은 MPEG 알고리즘에는 이동보상 보간 부호화 기법이 추가되었다.

3. 화면간 부호화(Interframe coding)

MPEG 부호화기법에서 화면간 부호화는 이동량 예측을 기본으로, P 화면과 B 화면에 적용된다. 화면간 부호화시 P 화면은 이전화면을 대상으로 매크로블럭의 16×16 크기의 휘도 성분만을 이용하여 이동량을 검출하고, B 화면에는 그림 4와 같이 이전화면과 이후화면을 기준으로 각각 이동량을 검출하여 현 매크로블럭을 예측하는 이동보상 보간 부호화(Motion Compensated Interpolative Coding)가 추가되었다. B 화면의 부호화는 단방향 예측의 P 화면의 부호화에 비교하여 보다 근사된 현재화면의 매크로블럭을 예측을 할 수 있고, 이전의 기준화면만으로 예측할 수 없는 영역(uncovered area)을 이후의 기준화면 또는 양방향 예측으로 적절히 처리할 수 있다.

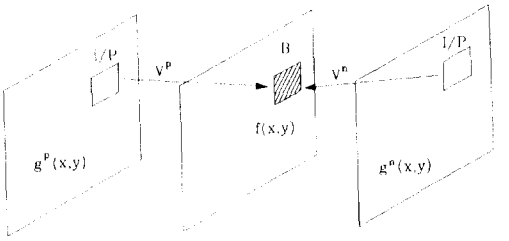


그림 4. 이동보상 보간 부호화
Fig. 4. Motion compensated interpolative coding.

화면간 부호화에서 매크로블럭을 부호화하기 위하여 이동량검출에 의한 예측 부호화 기법만을 이용하지 않고 연구 I의 두 가지 시험을 거쳐서 P 화면의 매크로블럭의 부호화 형태를 결정한다. B화면에서는 위의 두 가지 시험 이전에 이동량 검출을 위하여 순방향(forward), 역방향(backward)과 양방향(bidirectional) 예측을 수행할 것인지를 먼저 결정한다.

위의 기법을 이용한 화면간 부호화는 화면내 부호화 기법에 비교하여 적은 비트량을 소요하여, 큰 압축효율을 얻을 수 있다.

Ⅲ. 병렬처리 기법

연구 가에서 제안된 프로세싱 모듈 하나만으로 고속 동영상처리가 불가능하므로, 병렬처리기법을 도입하여 부호화를 수행하고자 한다. 따라서 본 장에서는 병렬처리시에 요구되는 기본적인 내용과 동영상 부호화에 적합한 병렬처리 기법에 관하여 논하였다.

1. 병렬처리 모델

동영상 부호화를 병렬처리하기 위해서, 다중 프로세서 시스템의 성능에 영향을 미치는 요소와 제안된 프로세싱 모듈로 구성이 가능한 병렬처리 모델을 고려하여야 한다. 먼저 멀티프로세서 시스템의 성능에 영향을 주는 요소를 고려하면 다음과 같다. [9]

1) 프로세서에 문제를 할당하는 방법
2) 각 프로세서에서 수행되는 작업의 크기(grain size)
3) 데이터 통신시 연산과 통신의 중복성(communication overlap)
4) 주 프로세서의 메모리에서 부 프로세서의 메모리에 데이터 전달 방법
5) 부 프로세서간의 연결방법과 통신속도
6) 부 프로세서의 연산능력과 메모리용량

먼저, 병렬처리기법의 분류에서 가장 기본적인 사항인, 프로세서에 문제를 분할하는 방법에는 파이프라인 기법과 분할 기법으로 나누어 생각할 수 있다. 그림 5 (a)와 (b)는 파이프라인 기법과 분할 기법의 작업할당의 예이다.

각 프로세서에서 수행되는 작업의 크기는 large grain과 fine grain으로 나누어 생각할 수 있고, 일반적으로 주어진 문제에 대하여 작업크기가 작아짐에 따라서 프로세서의 갯수가 증가한다. 따라서 fine grain 병렬처리는 단순한 작업에 효율적이고, large grain 병렬처리는 알고리즘과 제어가 복잡한 경우 유용하다.

연산과 통신의 중복성은 프로세서간의 통신을 위해서 소요되는 동작이 연산과 중복하여 수행이 가능한

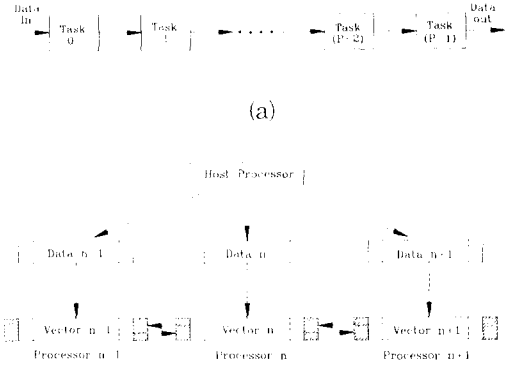


그림 5. 병렬 처리의 작업할당의 예
(a) 파이프라인 작업 할당
(b) 데이터 분할 기법의 작업 할당의 예

Fig. 5. Examples of the task allocation in parallel processing.
(a) The task allocation in pipeline processing,
(b) The task allocation in geometrical processing.

가 이다. 수행이 가능한 경우 overlapped communication mode이고, 가능하지 않은 경우에는 non-overlapped communication mode이다. 이 특성은 프로세서의 갯수가 적을 경우에는 시스템의 성능에 많은 영향을 준다.

주 프로세서의 메모리로부터 각 부 프로세서의 메모리로의 데이터 이동은 주 프로세서와 부 프로세서의 메모리를 공유하는 방법과 주 프로세서와 부 프로세서간의 직접적인 통신을 하는 방법이 있다.

프로세서의 연결방법은 시스템내에서 프로세서간에 데이터를 교환하기 위해 경로를 설정하기 위한 방법으로, 문제할당의 방법에 따라 많은 영향을 받는다. 본 연구에서는 4개의 20Mbps의 통신채널을 갖는 고성능 범용 프로세서, 각각의 프로세싱 모듈 내부에 대용량의 메모리와 특정 연산을 수행하는 ASIC으로 구성하였고, 이 모듈간의 연결을 자유롭게 할 수 있는 crossbar switch를 도입하여 구현한 시스템을 이용하여 동영상 부호화를 수행하고자 한다.

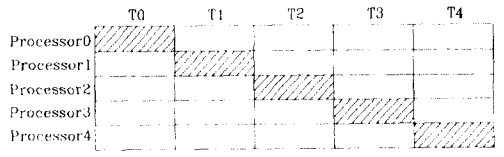
마지막으로 Transputer는 연산과 통신의 중복 수행이 가능한 프로세서이므로 이것을 고려하여 중복수행이 불가능한 경우와 더불어 동영상 부호화에 적합한 병렬처리 구조를 분석하였다.

2. 파이프라인 기법

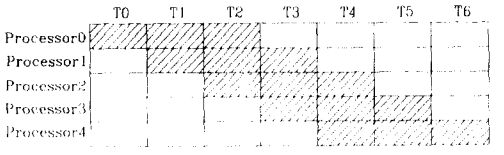
파이프라인기법은 전체 알고리즘을 여러 부분으로 나누어서 각각을 프로세서에 할당하여 작업을 수행하는 기법이다. 즉, 주어진 데이터 D에 대하여 알고리즘 A를 수행하는 경우, 파이프라인 기법은 A를 (A_v, A_e)로 생각할 수 있다. 이 경우 A_v는 부분 알고리즘에 관계되는 연산이 수행되는 노드이고, A_e는 각 노드간을 연결하는 경로이다.[5]

파이프라인기법에서 작업할당은 주어진 알고리즘을 다중 프로세서 시스템에서 수행시간의 균등성 확보와 최소의 통신비용을 갖도록 하는 것이 목적이다. 위의 두 사항은 다중 프로세서의 효율에 영향을 크게 주며, 그 외에 첫 결과를 얻기 위하여 파이프라인을 채우거나 또는 마지막 결과를 얻기 위하여 파이프라인의 내용을 인출하는 초과비용이 효율에 영향을 미친다.[7]

그림 6(a)는 각기 수행시간이 동일한 파이프라인 구조에서 한 개의 데이터 패킷만을 처리하는 예이다. 빗금친 부분은 프로세서가 작업을 수행하는 예이고, 공백 부분은 작업을 수행하지 않는 부분이다. 이 경우에는 다섯 개의 프로세서 중에서 하나만이 동작하므로 효율은 20%이다. (효율 = 5/25 = 0.2) 이 경우 프로세서의 갯수가 P인 경우 효율은 1/P로 감소한



(a)



(b)

그림 6. 5 프로세서에서의 파이프라인 효율
(a) 한 개의 패킷에 대한 파이프라인 활동도
(b) 세 개의 패킷에 대한 파이프라인 활동도

Fig. 6. The efficiencies of pipelining using five processors.
(a) The activity diagram for one data packet.
(b) The activity diagram for three data packets.

다. 그림 6(b)는 위의 같은 데이터를 3개의 작은 데이터 패킷으로 나누어 실행한 경우 전체 효율은 43% 이다. (효율 = 15/35 = 0.43) 위의 두 경우에서 파이프라인을 채우거나, 최종결과를 인출하는 오버헤드는 프로세서당 (P-1)주기가 필요하고 파이프라인 전체에 대해서는 P×(P-1)주기가 필요하다.

프로세서 갯수 P, 데이터패킷 갯수 N인 경우 전체 수행시간 T_{total}은

$$T_{total} = [N+(P-1)] \times T_{slow} \quad (1)$$

이나, 일반적으로 N ≫ P 이므로 T_{total}은

$$T_{total} = N \times T_{slow} \quad (2)$$

이다. 연산과 통신이 non-overlapped mode인 경우에 k 번째 파이프라인 단계에서의 연산시간과 통신시간을 각각 T_{proc_k}와 T_{comm_k}라고 정하면 T_{slow}는

$$T_{slow} = MAX\left(T_{pipe\_0}, T_{pipe\_1}, \ldots T_{pipe\_k}, \ldots T_{pipe\_(p-1)}\right)$$
$$T_{pipe\_k} = T_{pipe\_k} + T_{comm\_k} \quad (3)$$

이고, 연산과 통신이 overlapped mode인 경우에는 T_{total}은 non-overlapped mode와 동일하고, T_{slow}만이 아래와 같이 주어진다.

$$T_{slow} = MAX\left(T_{pipe\_0}, T_{pipe\_1}, \ldots T_{pipe\_k}, \ldots T_{pipe\_(p-1)}\right)$$
$$T_{pipe\_k} = MAX\left(T_{pipe\_k}, T_{comm\_k}\right) \quad (4)$$

위에서 T_{slow}는 파이프라인기법을 적용한 병렬처리 시스템에서 수행시간이 가장 많이 소요되는 프로세서의 수행시간이다.

한 개의 프로세서의 전체 수행시간이 T_{one}일 경우 전체 효율 E는

$$E = \frac{T_{one}}{P \times T_{total}} \quad (5)$$

이다.

3. 데이터 분할기법

데이터 분할기법은 처리하고자하는 전체 데이터를 시스템내의 프로세서들에게 적절히 나누어서 분배하고, 처리된 부분결과를 조합하여 전체 문제를 해결하는 병렬처리기법이다. 이 경우 각 프로세서에서 수행되는 알고리즘은 같고, 처리하는 데이터만이 다르다.

주어진 데이터 D에 대하여 알고리즘 A를 수행할 때, 데이터 D를 서로 중복되지 않는 데이터의 일부분 d_i로 나눌 경우 전체 데이터 D를 $D = \bigcup_i d_i$로 나타내면, 데이터 분할기법을 (A, d_i)의 쌍으로 표현할 수 있다.[5]

분할기법에서 계산량을 고려하면, P개의 프로세서상에서 k 번째 프로세서에서의 연산시간, 통신시간이 각각 T_{proc_k}와 T_{comm_k}이고, 수행되는 알고리즘과 데이터의 양이 동일하여 연산시간과 통신시간이 같은 경우 non-overlapped mode의 전체 연산시간 T_{total}은

$$T_{total} = T_{reg\_0} = T_{reg\_1} = \ldots = T_{reg\_k} = \ldots = T_{reg\_(p-1)}$$
$$T_{reg\_k} = \frac{T_{one}}{P} + T_{comm} \quad (6)$$

이다. 이 경우 효율 E는 통신에 대한 고려를 제외하면, 프로세서 갯수에 비례하여 증가한다. 그러나 P개의 프로세서상에서 처리되는 데이터의 양이 일정하지 않아서 연산시간이 모두 다른 경우 전체 연산시간 T_{total}은

$$T_{total} = MAX\left(T_{reg\_0}, T_{reg\_1}, \ldots T_{reg\_k}, \ldots T_{reg\_(p-1)}\right)$$
$$T_{reg\_k} = T_{proc\_k} + T_{comm\_k} \quad (7)$$

이고, 연산과 통신이 overlapped mode 경우의 전체 수행시간 T_{total}은

$$T_{total} = MAX\left(T_{reg\_0}, T_{reg\_1}, \ldots T_{reg\_k}, \ldots T_{reg\_(p-1)}\right)$$
$$T_{reg\_k} = MAX\left(T_{proc\_k}, T_{comm\_k}\right) \quad (8)$$

이다. 위에서 $T_{reg\ k}$는 분할기법을 적용한 병렬처리 시스템에서 k 번째 영역에서의 소요되는 프로세서의 수행시간이다.

분할기법에서의 전체 효율 E는

$$E = \frac{T_{one}}{P \times T_{total}} \quad (9)$$

이다.

Ⅳ. 병렬 동영상 부호화

본장에서는 MPEG 동영상 부호화기법을 3장에서 언급한 병렬처리 구조에 적용시켜 타당성을 고려한다. 실제로 MPEG에는 동영상 뿐 아니라 정지영상 부호화에 관한 내용도 포함하고 있으나, 본 장에서는 실제 계산량이 많고, 계산의 변화를 많이 포함한 동영상 부분만을 고려한다. 처리되는 알고리즘의 작업 할당은 그림 7과 같다. [11]

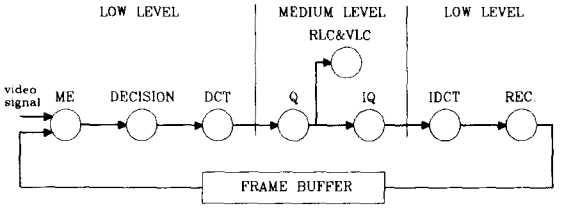


그림 7. 동영상 부호화 작업 할당

Fig. 7. The allocation of moving picture coding tasks.

그림 7의 동영상 부호화 알고리즘을 단일 프로세서 상에서 구현하는 경우의 계산량은 다음과 같다.

$$T_{one} = \left(T_{ME} + T_{DECL} + T_{DCT} + T_Q + \left(T_{IQ} + T_{IDCT}\right) \times (1 - SF)\right) \times N \quad (10)$$

T_{ME} : *motion estimation time*$\left(= forward\_f\_code \times T_{ME16}\right)$

T_{DECL} : *MC/noMC*, int *ra*/non int *ra decision time*

$T_{(I)DCT}$: *macroblock (I)DCT time*$\left(= 4 \times T_{(I)DCT\_interer}\right)$

$T_{(I)Q}$: *macroblock (I)Quantization time*$\left(= 4 \times T_{(I)Q}\right)$

SF : *Skip Factor*$\left(= 1 - \frac{Number\ of\ coded\ blocks}{Number\ of\ blocks\ in\ picture}\right)$

N : *Number of macroblocks in the picture*

T_{ME}는 화면의 이전화면과 현화면의 간격을 반영하여 forward_f_code를 도입하였다. [3] $T_{(I)DCT}$, $T_{(I)Q}$의 계산량에서 숫자 4는 매크로블럭에 포함된 블럭중 휘도 성분만을 고려함이다. SF는 양자화후의 계수가 모두 영 값이면 역양자화 및 IDCT연산이 생략되는 것을 반영하기 위함이고, N은 처리되는 데이터 패킷의 수로 이 경우에는 MPEG SIF영상에서 처리되고자하는 매크로블럭의 수이다. 본 연구에서는 MPEG 전체 알고리즘중에서 RLC와 VLC는 주 프로세서에서 수행하고, 부 프로세서에서는 나머지 부분을 수행한다.

1. 파이프라인 동영상 처리

동영상 알고리즘을 파이프라인 구조에 적용시 가장 중요한 사항은 그림 7에 나타난 작업그래프를 부 프로세서에 적절히 할당하는 것이며, 이것이 파이프라인기법을 채택한 시스템에서 성능에 가장 큰 영향을 주는 요소이다.

본 연구에서는 3장에서 구한 각 부분 알고리즘이 갖는 작업량의 불균등과 통신량을 최소로 하는 것을 목적으로 프로세서에 작업을 할당하였다. 그림 7의 작업 그래프를 그림 8과 같이 프로세서 3개에 적용하였다.

동영상 부호화기법을 파이프라인기법으로 구현시에 효율을 저하시키는 요인은 각 부분 알고리즘의 계산량 불일치와 함께, 이전화면의 예측오차의 크기에 따른 계산량의 큰 변화를 들 수 있다. 이 사항은 동영상 부호화에서 이동량을 검출하여 현화면과 이전화면과의 차 영상에 대하여 DCT를 수행하여 양자화한 결과 모든 계수가 영 값이면 양자화 이후의 작업이 생략된다. 이러한 경우 양자화이후의 작업을 처리하는 프로세서는 작업을 수행하지 않으므로 전체 효율이 크게 저하된다.

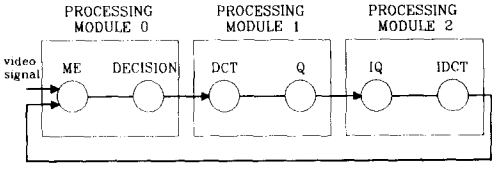


그림 8. 동영상 부호화 알고리즘의 파이프라인 적용

Fig. 8. An application of pipelining on moving picture coding algorithm.

그림 8과 같이 작업이 할당된 경우 각 프로세싱 모듈에서의 수행시간은 식(11)과 같다.

$$T_{proc\_0} = T_{ME} + T_{DECL}$$
$$T_{proc\_1} = T_{DCT} + T_{Q}$$
$$T_{proc\_2} = \left(T_{IQ} + T_{IDCT}\right) \times (1 - SF) \quad (11)$$

그림 8과 같이 작업이 할당된 경우 각 프로세싱 모듈에서의 통신시간은 식(12)와 같다.

$$T_{comm\_0} = \left(MBS^2 \cdot (1 - SF) + 2 \cdot MBS^2\right) \times \beta$$
$$T_{comm\_1} = \left(2 \cdot MBS^2 + (1 + s) \cdot MBS^2\right) \times \beta$$
$$T_{comm\_2} = \left((1 + s) \cdot MBS^2 + MBS^2 \cdot (1 - SF)\right) \times \beta \quad (12)$$
$$where, \beta = \frac{1\ byte\ transmission\ time}{processor\ cycle\ time}$$

위의 식에서 MBS^2은 매크로블럭내의 휘도 성분의 화소 갯수로 256개이고, s는 양자화 계수당 byte 크기로 2 이고, β 는 통신에 관계되는 시간을 연산시간에 정량적으로 반영하기 위함이다. 위의 식 (12)의 첫 항은 데이터 입력에 관한 것이고, 둘째는 출력에 관한 항이다.

3장에서 구한 부분 알고리즘의 연산량과 식 (12)의 통신량을 대입하여 전체 수행시간을 각각에 대하여 구하면 다음과 같다.

$$non-overlapped\ mode: T_{total} = \left(T_{proc\_0} + T_{comm\_0}\right) \times N$$
$$overlapped\ mode: T_{total} = T_{proc\_0} \times N \quad (13)$$

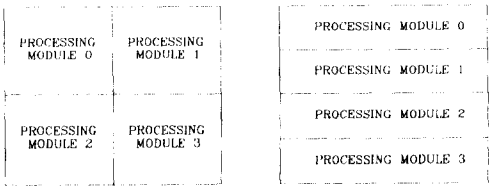
마지막으로 MPEG 영상 부호화를 파이프라인기법에 적용할 때 계산의 불일치외에 발생되는 중요한 문제점은 I 화면과 P 화면, B 화면의 각각에 대해 서로 다른 부호화기법을 이용하는 점이다. 이 화면 부호화기법은 각기 수행하는 연산이 다르다. 즉, I 화면은 이동량 검출에 관한 연산이 완전히 배제되고, B 화면은 양쪽 방향을 기준으로 이동량을 검출하기 때문에 크기가 적은 차영상이 발생되고, 또한 발생되는 비트를 줄이기 위하여 양자화 단계수를 크게 하기 때문에 부호화되는 블럭이 P 화면에 비교하여 크게 감소한다. 따라서 이와 같은 문제점 때문에 발생되는 계산량의 불일치로 인해 파이프라인기법은 복합 영상 부호화기법을 실현하는데 부적합하다.

2. 분할기법 동영상 처리

분할기법은 시스템내에 주 프로세서만을 제외하고, 모든 부 프로세서가 전체 처리하고자 하는 데이터중에서 일부분에 대하여 동일한 알고리즘을 수행하고, 처리한 결과를 다시 주 프로세서에 전달한다.

분할기법으로 병렬처리를 수행시 프로세서에 데이터 할당에는 그림 9와 같이 크게 두 가지가 있다.[7]

동영상 부호화에 두가지 할당을 고려하면, stripwise 분할법에서는 프로세서 갯수 P가 8의 경우에는, 화면간 부호화 같이 현화면 처리시에 이전화면의 데이터가 많이 필요한 경우 이웃한 프로세서의 메모리에 데이터가 존재하지 않으므로 통신량이 급격하게 증가하고, 또한 P가 15 이상의 경우에는 한 프로세서가 담당하는 열(row)이 이동량 검출단위인 매크로블럭의 크기 16 보다 작아 동영상 부호화가 불가능하다. 따라서 본 연구에서는 이러한 동영상 부호화시 통신의 효율을 고려하여 boxwise 분할기법을 이용한다.



(a) (b)

그림 9. 영상 데이터의 프로세서 할당
(a) boxwise 할당법
(b) stripwise 할당법

Fig. 9. The allocation of image data for the processors.
(a) A boxwise allocation,
(b) A stripwise allocation.

Boxwise 할당에서 프로세서의 갯수 P는 2×2^{2n}(P = 2, 4, 8, 16, … for n=0, 1, 2, 3, ..)이며, x 방향의 프로세서의 갯수 P_x, y 방향의 프로세서의 갯수 P_y라고 하면 $P = P_x \times P_y$이다.

분할기법에서 통신은 다음 세가지 요소로 나누어서 생각할 수 있다.

$$T_{comm\_k} = T_{DP} + T_{PI} + T_{OVR} \quad (14)$$
T_{DP} : *data partition time*
T_{PI} : *partial results integration time*
T_{OVR} : *overlapped data interchange time*

T_{DP}와 T_{PI}는 통신량은 약간 차이가 있으나, 통신방법은 유사하고, 방향만이 반대이다. 즉, T_{DP}는 처리하고자하는 352×240 크기의 MPEG SIF 영상데이터를 프로세서에게 분할시키는 시간이고, T_{PI}는 처리된 영상의 양자화된 값과 매크로블럭에 관한 정보를 master 프로세서에 전달한다. T_{PI}는 두개의 부분

결과만이 합하여 새로운 부분결과를 발생시키는 작업을 반복하여 수행하고 다음과 같다.

$$\begin{aligned} T_{PI} &= \left[(D_o/P)+2(D_o/P)+4(D_o/P)+ \right. \\ &\qquad \left. \cdots +2^{(k+1)}(D_o/P)\right]\times\beta \\ &= \beta(D_o/P)\cdot\left[1+2^1+2^2+\cdots+2^{(k+1)}\right] \\ &= \beta(D_o/P)\cdot(2^k-1) \end{aligned} \tag{15}$$

여기에서 D_o는 전체 출력 데이터의 크기이고, k는 통신 단계를 나타내는 파라메터로 프로세서의 갯수와 연관이 있다. 즉, P개의 부 프로세서에 나누어진 부분결과가 k 통신단계를 거쳐 주 프로세서에 집중된다. 따라서 k는 프로세서 갯수 P에 따른 전체 통신 단계 수이다.

$$\begin{aligned} &P/2 \textit{ partial results after completion of } \text{interval1} \\ &P/4\left(=P/2^2\right)\textit{partial results after completion of } \text{interval2} \\ &\qquad \cdots \\ &P/2^k \textit{ partial results after completion of } \text{interval}k \\ &P/2^k+1, k=\log_2 P \end{aligned} \tag{16}$$

식 (16)의 k를 식 (15)에 대입하면, T_{PI}는 식 (17)과 같고, 데이터 결합 순서는 ALGORITHM 1과 같다.

$$T_{PI}=\beta(D_o/P)(P-1) \tag{17}$$

ALGORITHM 1 PARTIAL RESULTS SUM ALGORITHM

```
For i=0 to log2P do
  For all processing modules j do in parallel
  ( 0≤j ≤P 1 )
    If j mod 2^(i+1) = 2^i then
      Connect Pj -> Pj-2^i
      Send message
    End If
  End_for
End_for
```

위의 알고리즘에서 프로세싱 모듈 번호는 연구 I의 그림 11의 (a)와 같이 행간행(row by row)으로 부여된다. T_{DP}는 부분 데이터를 전달받는 경우에 대한 고려이고, T_{PI}는 처리하고자 하는 데이터를 각 프로세서에 전달하고자 하는 과정이다. 따라서 통신의 방향은 반대이지만 통신량 계산 과정은 일치한다. 따라서 T_{DP}를 구할 때, 위에 전개한 수식을 그대로 이용이 가능하다. 그러나 위의 T_{DP}와 T_{PI}에서 각각 주 프로세서와 부 프로세싱 모듈간의 데이터 교환은 파이프라인 기법과의 비교를 위하여 고려하지 않았다.

분할된 화면의 경계부분에서 이동량을 검출하고자 하는 경우, 이웃한 프로세서에서 처리된 이전화면의 데이터가 필요하다. T_{OI}는 이러한 통신에 소요되는 통신량이다. 이 통신량은 이동량의 범위에 따라 크게 변화한다. MPEG 부호화의 경우 각 화면의 압축된 데이터 앞부분에 그 화면의 forward_f_code, backward_f_code와 full_pel 이동량의 범위를 부호화한다. 표 1은 위 두 변수에 따른 이동 벡터의 범위를 나타낸다. [3]

표 1. 이동벡터 범위

Table 1. The range of the motion vectors.

forward_f_code or backward_f_code	Motion vector range(MVR)	
	full_pel=0	full_pel=1
1	-8 to 7.5(8)	-16 to 15(16)
2	-16 to 15.5(16)	-32 to 31(32)
3	-32 to 31.5(32)	-64 to 63(64)
4	-64 to 63.5(64)	-128 to 127(128)

P_x와 P_y에 의해 분할된 영상의 크기가 $N_x \times N_y$의 경우, T_{OVR}은

$$\begin{aligned} T_{OVR} &= \sum T_H + \sum T_V + \sum T_R \\ T_H &= N_x \times MVR \times \beta \\ T_V &= N_y \times MVR \times \beta \\ T_R &= MVR \times MVR \times \beta \end{aligned} \tag{18}$$

위의 식에서 T_H, T_V, T_R은 복원된 이전화면을 기준으로 이동량 검출을 수행하는 경우 자신의 메모리에 존재하지 않는 수평과 수직, 모서리 부분의 복호화된 영상을 이웃한 프로세서에 전달하기 위한 통신량이다.

본 연구에서는 부호화기의 성능에 큰 영향을 주지 않는 모서리 부분에서의 이동량검출은 생략한다. 따라서 모서리부분의 통신 T_R은 생략한다.

$$T_{OVR}+\sum T_H \sum T_V \tag{19}$$

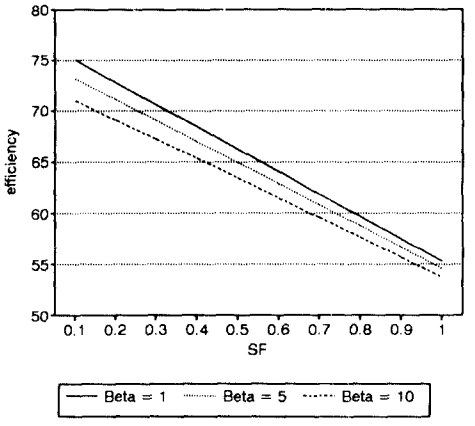
프로세싱 모듈 k가 처리하는 데이터 패킷의 갯수를 N_k, 처리하는 영역의 skip factor를 SF_k 라고 정의할 때, 동영상 부호화에서 $T_{reg\ k}$는

$$\begin{aligned} T_{reg\ k} &= \left[T_{MF}T_{DECL}T_{DCT}+T_Q+\left(T_{IQ}+T_{IDCT}\right)\times(1-SF_k)\right]\times N_k \\ SF_k &: \textit{Skip Factor in } k-th \textit{ region} \\ N_k &: \textit{Number of packets to be processed in } k-th \textit{ region} \end{aligned} \tag{20}$$

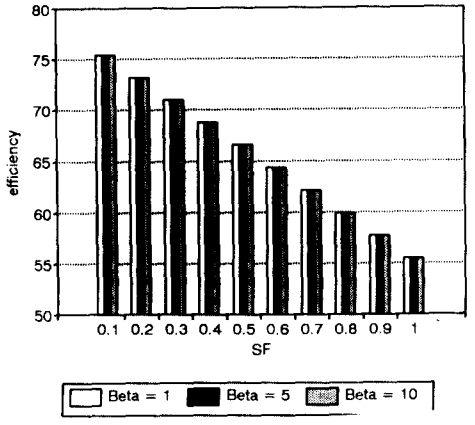
이다. 위의 식에서 보듯이 분할기법에서의 알고리즘 수행시간은 처리하고자하는 데이터의 패킷의 수가 일정한 경우, SF_k에 많은 영향을 받는다. 가장 적은 SF_k를 갖는 영역의 처리시간이 전체 수행시간이다.

3. 성능 평가

동영상 부호화기법을 파이프라인 기법과 지형적 분할기법으로 처리시에, β와 SF를 변화시키면서 효율을 구하여 성능을 비교하고자 한다. 이 경우 β가 적은 경우에는 통신을 고속으로 수행할 수 있는 시스템이고, β가 증가함에 따라 통신의 오버헤드가 큰 시스템이다.



(a)



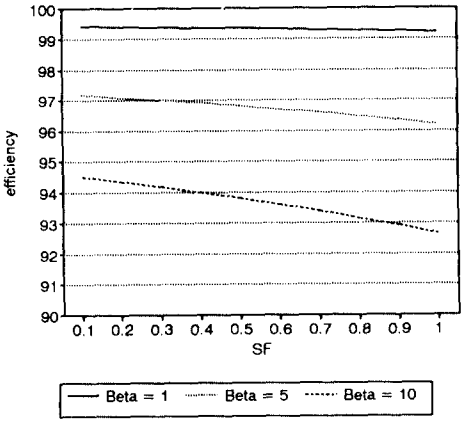
(b)

그림 10. 파이프라인 구조의 효율
Fig. 10. The efficiency of the pipeline structure. (a) non overlapped, (b) overlapped.

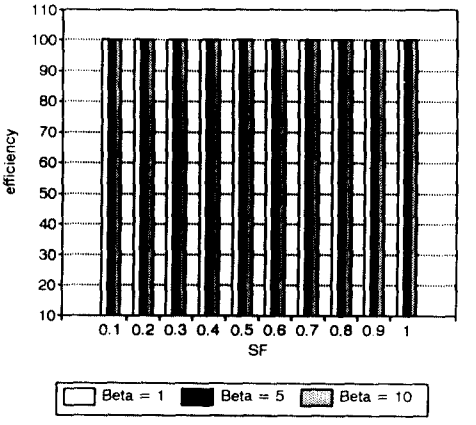
파이프라인 기법에서는 그림 8과 같이 작업이 할당된 경우, 연산시간과 통신시간을 구하여 non-overlapped와 overlapped mode에 대하여 효율을 구하면 그림 10과 같다.

위의 그림 10 (b)의 overlapped 형태에서 β와 무관하게 효율이 동일한 것은 통신량이 계산량에 비교하여 상대적으로 적은 값을 나타내고 있기 때문이고, β가 20 정도에서는 전체 효율이 통신에 의하여 감소한다.

지형적 분할 기법에서는 boxwise 분할을 기본으로



(a)



(b)

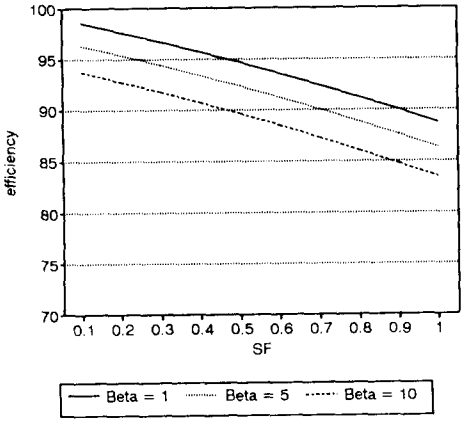
그림 11. 지형적 분할기법의 효율(SFk : 균일)
Fig. 11. The efficiency of the geometric allocation(SFk : uniform). (a) non overlapped, (b) overlapped.

각 영역의 SFk를 동일하다고 가정하고, 프로세서 4개에 알고리즘을 적용하여 효율을 구하였다.

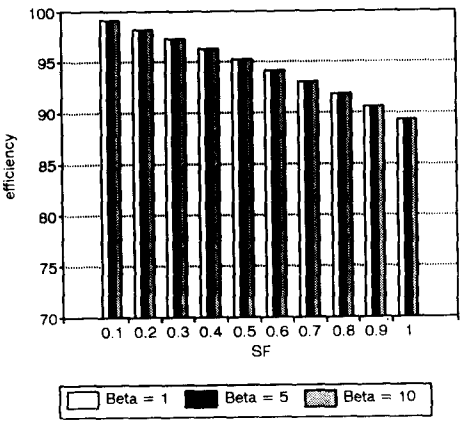
이 경우 수행시간은 식(6) $T_{one}/4$와 같고, 통신량은 식(14)에서 파이프라인의 경우와 같이 결과를 모으는 T_{in}를 제외하고 구하여 non-overlapped와 overlapped 형태에 대하여 효율을 구하면 그림 11과 같다.

위의 그림 11 (b)의 overlapped 형태에서의 결과는 위의 파이프라인 기법에서와 동일하게 통신에 따른 수행시간의 영향이 없음을 나타낸다. 그러나 위의 결과는 분할된 영역에서 SF_k가 모두 같다고 가정하여 효율을 구하였으나, 실제 부호화시에는 영역에 따라 SF_k가 다르다. 따라서 SF_k가 가장 적은 영역의 부호화가 끝나야, 전체 부호화가 종료되므로 전체 효율의 저하를 가져온다. 그림 12와 그림 13은 4개의 영역중에서 가장 작은 SFk가 전체 평균 SF보다 30%와 60% 작은 경우의 고려이다.

지형적 분할기법에서 모든 영역에 대하여 SF_k가 일정하지 않은 경우의 전체 부호화시간은 SF_k가 동일한 경우에 비교하여 약 10-20% 정도의 효율 저하를 가져온다.

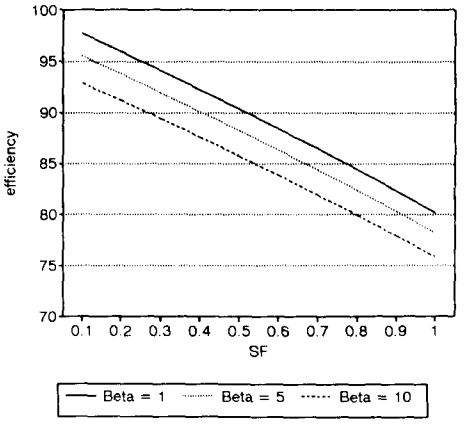


(a)

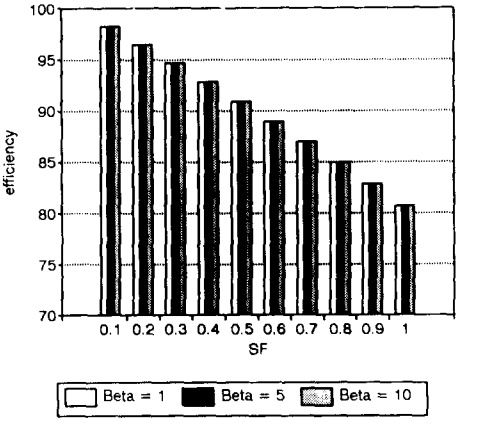


(b)

그림 12. 지형적 분할기법의 효율(SFk:30% 평균 이하)
Fig. 12. The efficiency of the geometric allocation (SFk: 30% below average). (a) non overlapped, (b) overlapped.



(a)



(b)

그림 13. 지형적 분할기법의 효율(SFk : 60% 평균 이하)
Fig. 13. The Efficiency of the geometric allocation (SFk : 60% below average). (a) non overlapped, (b) overlapped.

위의 결과를 통하여 파이프라인 기법과 분할 기법을 비교하면, 먼저 파이프라인 기법은 프로세서의 갯수를 임의로 증가하는 경우 큰 효율 저하를 가져올 수 있고, 작업을 계산량에 맞게 할당하여도 가장 수행시간이 많이 소요되는 프로세서에 의하여 성능이 제한된다. 또한, SF의 증가에 따라 효율 향상이 전혀 없다.

따라서 알고리즘 수정과 확장시의 장점과 SF_k가 일정하지 않은 경우에도 전체적인 효율이 파이프라인 기법보다 우수한 지형적 분할 기법이 동영상 부호화기에서 적합하다.

V. 실험 및 결과고찰

본 연구에서 구현한 멀티프로세서 시스템의 성능평가를 위하여 MPEG 알고리즘을 지형적 분할기법과 파이프라인 기법을 각각 수행하여 효율과 가속성을 구하였다. 병렬처리기법을 통한 알고리즘 수행에 앞서 352×240 크기의 Football 영상 150장을 각각 SUN-SPARC STATION II 와 연구 I에서 제안한 프로세싱 모듈 한개만을 이용하여 N=15, M=1인 MPEG 알고리즘을 수행하여 그림 14에 표시하였다. 모든 기법의 공정한 비교를 위하여 양자화 단계수는 모든 화면에 걸쳐 15로 동일하게 결정하여 SNR과 출력 비트량을 동일하게 하였다.

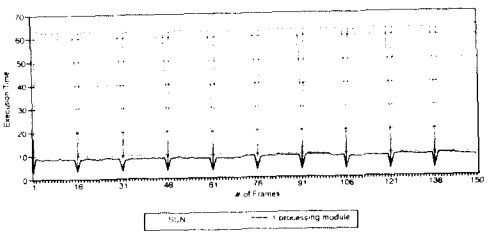


그림 14. 단일 프로세서상에서 MPEG 알고리즘 수행

Fig. 14. MPEG Algorithm execution using one processor.

위의 결과로부터 제안한 프로세싱 모듈을 통한 알고리즘 수행이 수행속도에서 7배 정도의 우수함을 보였다. 위의 속도 향상은 실제로 매크로블럭의 부호화 형태 결정과 양자화등이 소요되는 연산이 진행되는 Transputer의 성능이 SUN에 비교하여 떨어져서 DCT와 이동량 검출부에서 얻은 만큼의 성능 향상은 이루지 못하였다.

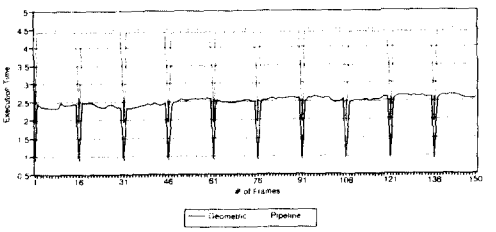


그림 15. 병렬처리기법의 수행시간 비교

Fig. 15. The comparision of parallel processing time.

그림 15는 파이프라인 기법과 지형적 분할 기법을 통하여 부호화 알고리즘을 수행한 시간 비교이고, 그림 16에 효율을 도시하였다.

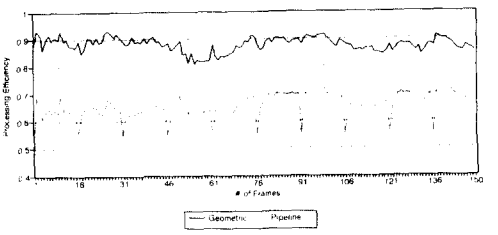


그림 16. 병렬처리기법의 효율 비교

Fig. 16. The comparision of parallel processing efficiency.

파이프라인기법의 효율저하는 첫째 이동량 검출과 매크로블럭의 부호화 형태를 결정하는 작업을 수행하는 프로세싱 모듈 0가 다른 모듈에 비교하여 큰 수행시간을 갖고, 둘째는 양자화 이후의 계수가 모두 0인 경우 역양자화와 IDCT가 생략되므로 프로세싱 모듈 2가 적절하게 활용되지 못하기 때문이다. 즉, 알고리즘 내부에 포함하고 있는 계산량의 불일치와 1 보다 작은 SF_k에 따른 잇점을 이용할 수 없으므로 효율이 크게 감소한다.

지형적 분할기법의 효율 저하는 첫째는 이동량 검출 단위인 매크로블럭 단위의 화면을 분할함에 따라 프로세싱 모듈 0과 1은 176×128 크기의 부 영상을 처리하고 프로세싱 모듈 2와 3은 176×112를 처리하여, 프로세싱 모듈간의 계산량이 차이가 발생하고, 둘째는 영역별 SF_k의 불일치에 따라 역양자화후에 처리되는 계산량이 각각의 모듈에 대하여 다르기 때문이다.

마지막으로 분할기법에서의 영역별 SF_k의 불일치

와 적응적 분할기법의 전체 수행시간과의 분석을 위하여 Football 영상의 각 영역별 부호화시간과 SF_k를 그림 17과 그림 18에 도시하였다.

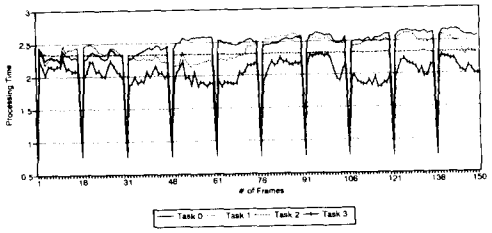


그림 17. Football 영상의 영역별 부호화시간
Fig. 17. The regional encoding time of the Football sequence.

위의 그림에서 매크로블럭 단위로 영상을 분할함에 따라 176×128 크기의 부 영상을 처리하는 프로세싱 모듈 0과 1이 176×112를 처리하는 프로세싱 모듈 2와 3에 비교하여 수행시간이 더 소요됨을 알 수 있고, 영역별 부호화시간과 SF_k와의 밀접한 관계가 있음을 알 수 있다.

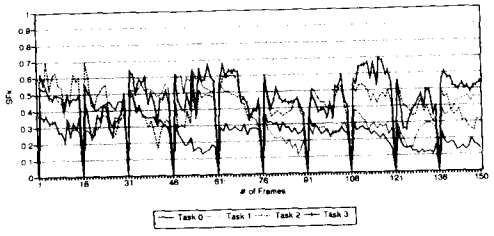


그림 18. Football 영상의 영역별 SF_k
Fig. 18. The regional SF_k of the Football sequence.

표 2는 Football 150장 영상에 대하여 전체 부호화 평균시간, 전체 효율과 속도 향상을 구한 것이다.

표 2. 구현된 시스템의 성능
Table 2. The performance of the implemented system.

	SUN	single processing	pipeline processing	geometric processing
time (sec)	58.89	8.46	4.259	2.408
efficiency (%)	-	100	65.8	87.9
speed up	-	1	1.98	3.51

마지막으로 제안한 부호화기를 이용하여 Football 영상을 부호화한 후, PSNR(Peak to peak signal to noise)과 출력 비트량을 그림 19와 그림 20에 도시하였다.

$$PSNR + 10 \cdot \log_{10}\left\{\frac{255^2}{E\left[(x-x')^2\right]}\right\} \tag{20}$$

x : origina image pixel value

x' : decoded image pixel value

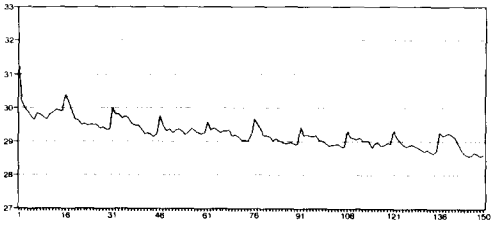


그림 19. 결과 PSNR
Fig. 19. Result PSNR.

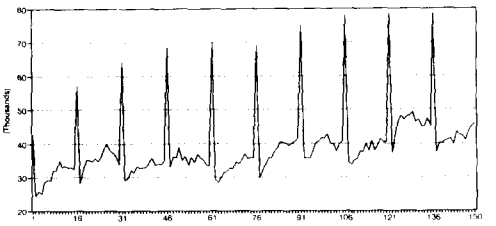


그림 20. 결과 출력 비트량
Fig. 20. Result output bit rate.

그림 21은 50, 100, 150번째의 복호화된 영상과 오차영상이다. 오차영상은 실제의 크기에 절대값을 취하여 6배를 곱하여 표시하였다.

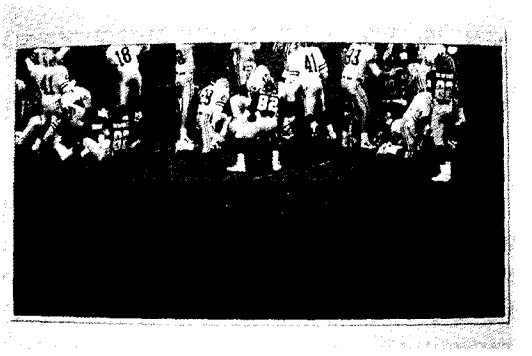


그림 21. 50, 100, 150 번째 복호화된 화면
Fig. 21. 50, 100 and 150th reconstructed images.

Ⅵ. 결론

연구 I에서 구현한 Transputer와 DCT, 이동량검출 프로세서를 결합하여 구성한 프로세싱 모듈을 기반으로 병렬처리 기법을 도입하여 동영상 복합부호화 장치를 구현하였다. 병렬처리기법 중에서 파이프라인 기법과 분할기법을 동영상 부호화기법에 적용할 경우 수행시간과 효율을 고려하였다.

결과고찰을 통하여 파이프라인기법은 MPEG 알고리즘과 같이 알고리즘 내용과 수행시간의 변화가 큰 경우에는 적합하지 못함을 보였다. 분할기법은 파이프라인 기법 보다는 우수한 성능을 입증하였으나, 분할된 영역마다 다른 계산량으로 인하여 프로세서의 증가에 따른 선형적인 성능 향상을 얻지 못하였다. 따라서 향후 계산량의 예측을 통한 적응적 부하균등(adaptive load balancing) 기법을 통하여 수행효율을 증대시키는 연구가 필요하다.

參 考 文 獻

[1] R. S. Cok, *Parallel Programs for the Transputer*, Prentice-Hall, 1990.

[2] ISO/IEC JTC1/SC2/WG11, "Coded Representation of Picture and Audio Information," ISO/IEC JTC1/SC2/WG11 N MPEG90/263, Sept. 1990.

[3] ISO/IEC JTC1/SC2/WG11, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbits/s," ISO/IEC JTC1/SC2/WG11, Nov. 1991.

[4] M. L. Liou, "Visual Telephony as an ISDN Apllication," IEEE Communications Magazine, pp. 30-38, Feb. 1990.

[5] S. Yalamanchili and J. K. Aggarwal, "Analysis of a Model for Parallel Image Processing," Pattern Recognition vol. 18 no. 1, pp. 1-16, 1985.

[6] S. Yalamanchili and J. K. Aggarwal, "A System Organization for Parallel Image Processing," Pattern Recognition vol. 18 no. 1, pp. 17-29, 1985.

[7] L. Ridgway Scott, "Load Balancing on Message Passing Architectures," Journal of Parallel and Distributed Computing 13, pp. 312-324, 1991.

[8] S. H. Bokhari, *Assignment Problems in Parallel and Distributed Computing*, Kluwer Academic Publishers, 1987

[9] Z. Cvetanovic, "The Effects of Problem Partitioning, Allocation, and Granularity on the Performance of Multiple-Processor Systems," *IEEE Trans.* on Computers, vol. c-36 no. 4, pp. 421-432, April 1987.

[10] S. H. Bokhari and Marsha J. Berger, "A Partitioning Strategy for Nonuniform Problems on Multiprocessor," *IEEE Trans.* on Computers, vol. c-36 no. 5, pp. 570-580, May 1987.

[11] P. Pirsch, et al, "Multiprocessor Performance for Real-Time Processing of Video Coding Applications," *IEEE Trans.* on CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, vol. 2 no. 2, June 1992.

[12] A. N. Choudhary, et al, *Parallel Architectures and Parallel Algorithms for Integrated Vision Systems*, Kluwer Academic Publishers, 1990.

著者紹介

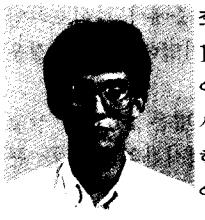
崔 翔 勛(正會員)

1965年 8月 27日生. 1987年 2月 아주대학교 전자공학과 졸업(공학사). 1989年 2月 연세대학교 전자공학과 졸업(공학석사). 1993年 8月 연세대학교 전자공학과 졸업(공학박사). 주관심분야는 영상압축, 컴퓨터 아키텍쳐 등임.

•

金 仁(正會員)

1969年 9月 12日生. 1992年 2月 연세대학교 전자공학과 졸업(공학사). 1992年 3月 ~ 현재 연세대학교 전자공학과 공학석사 재학중. 주관심분야는 영상압축 등임.



李 光 基(正會員)

1963年 7月 9日生. 1986年 2月 연세대학교 전자공학과 졸업(공학사). 1988年 8月 연세대학교 전자공학과 졸업(공학석사). 1993年 8月 연세대학교 전자공학과 졸업(공학박사). 주관심분야는 영상압축 등임.

•

李 鎔 均(正會員) 第 30卷 B編 第 2號 參照
현재 한국전자통신 연구소 ISDN 신호처리 연구실장

•

朴 圭 泰(正會員) 第 28卷 B編 第 11號 參照
현재 연세대학교 전자공학과 교수