구조적 표현의 화상 처리를 위한 ASIC 설계 연구
A Study on the Design of ASIC for the Images in the Hierarchical Representation

김 종완*, 이 기한, 김 경식, 황 희융
서울대학교 전자계산기공학과

<요약>

본 연구에서는 구조적 표현의 화상 처리 알고리즘인 BF(Breadth First) 선형 4진 트리 알고리즘(BFQT 알고리즘)의 압축, 재생부를 하드웨이화 하여 ASIC (Application Specific Integrated Circuit)을 설계한다. ASIC과 IBM PC와의 인터페이스를 명시하며, 새로운 하드웨이 알고리즘을 도입하여 ASIC의 세부구조를 설계한다. 소프트웨이로 수행할 때 보다 제안된 ASIC으로 수행할 때가 압축은 약 21배, 재생은 약 4배 빨라지는 것으로 추정된다.

I. 서론

ASIC(Application Specific Integrated Circuit)이란 응용 분야가 정해진 IC로서 현재 활발히 연구 개발되고 있는 주문형(custom) IC이다[1].

본 논문에서는 화상을 구조적으로 표현(hierarchical representation)하여 기억장소를 절감하고 화상 처리 알고리즘의 처리 시간을 단축시키기 위하여 화상 처리 알고리즘 중에서도 많은 시간이 소요되는 압축 및 재생 부분을 ASIC화 하려고 한다.

하드웨이화 할 화상처리 알고리즘으로서는 압축률이 우수하고 여러가지 화상 처리 연산이 용이한 BF(Breadth First) 선형 4진 트리 표현 방법을 택하였다. 이러한 알고리즘을 본 논문에서는 BFQT(BF QuadTree) 알고리즘이라 부르겠다.

본 연구는 하드웨어로 화상 처리 알고리즘을 구현하는 것으로서, 소프트웨어로 구현할 때보다 속도가 몇 배 빠르게 되고, 그에 따라서 X-ray, Videotex등 화상의 실시간 처리가 요구되는 곳에 이용 가능하다.

소프트웨어를 설계할 때와 마찬가지로 하드웨어를 설계할 경우에도 정보와 제어의 흐름을 체계적으로 고려하기 위하여 하드웨어 알고리즘이라는 개념을 도입한다. 이러한 하드웨어 알고리즘으로부터 바로 논리 회로 설계가 가능하므로 ASIC 설계를 용이하게 해 준다.

BFQT 알고리즘과 ASIC화 영역에 대해 II장에서 기술하고, IBM PC와의 인터페이스에 대해 III장에서 명시하며, ASIC의 기능 구조, 압축부와 재생부의 하드웨어 알고리즘, 결과 하드웨어 일부를 IV, V, VI장에서 각각 설명한다. 제안된 ASIC에 의한 시간 이익을 추정하여 VII장에서 평가하였다.

II. BFQT알고리즘과 ASIC화 영역

BFQT 알고리즘은 헤더부(header part)와 데이타부(data part)를 따로 두어 각 노드들 사이의 포인터(pointer) 정보를 없앤 점은 노드들의 위치 정보만 저장하므로 정규 4진 트리에 비하여 기억 장소면에서 효과적이다[2].

2^n X 2^n 해상도의 이진 화상을 BFQT 표현으로 변환하는 과정을 압축이라고 한다. 이진 화상을 압축하는 과정은 이진 화상의 축소 과정과 축소된 이진 화상으로부터 BFQT 표현을 유도하는 과정으로 나눌 수 있다.

재생 과정은 압축 과정의 역순으로서 이진 화상을 표현하고 있는 BFQT 표현으로부터 원래의 이진 화상을 얻는 과정이다.

본 논문은 1987년도 문교부 학술 연구 조성비에 의해 수행된 연구 결과임

본 연구는 BFQT 알고리즘[3] 중에서도 가장 기본적이면서 시간이 많이 소요되는 압축, 재생 부분을 ASIC[4]화 하며 중복, 공유, 반전, 회전, 이동, 분리, 면적 산출, 추출등의 연산 부분은 소프트웨어로 처리한다.

BFQT 알고리즘의 전체적인 기능 구조와 ASIC화 할 부분은 그림 1에 명시한다.

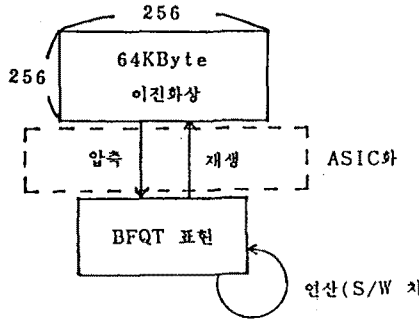


그림 1. BFQT 알고리즘의 기능 구조와 ASIC화 할 부분

III. IBM PC와 ASIC간의 시스템 인터페이스

ASIC의 입출력인 이진 화상 정보와 BFQT 표현은 ASIC 외부에 두 메모리 블럭을 두어 보관하며 이를 각각 BIN MEMORY, BFQT MEMORY라 하고 (그림 2 참고), ASIC과 두 메모리 블럭을 포함하여 ASIC 보드라 한다.

IBM PC 시스템과 ASIC 보드 사이의 인터페이스[5]는 소프트웨어로 처리한다(그림 2를 참조). 즉, PC에서 "압축"이라는 명령(예: BFQT_ENCODE)을 주면 인터페이스 소프트웨어가 PC내의 이진 화상 정보를 ASIC 보드내의 BIN MEMORY내에 적재시키고 ASIC에 "압축 인에이블" 신호를 주고 PC는 ASIC으로부터의 압축의 완료를 기다린다. ASIC에서는 압축 인에이블 신호를 받으면 BIN MEMORY의 화상 정보를 처리(encoding)하여 BFQT MEMORY에 BFQT 표현으로 적재시키고 압축 완료 인터럽트 신호를 발생시킨다. 그때 인터럽트를 기다리고 있던 인터페이스 소프트웨어가 BFQT MEMORY내의 내용을 BFQT 화일로 만들어 PC에 저장하고 실행을 종료한다.

재생 명령의 경우(예: BFQT_DECODE)에도 이와 유사하며 그림 2에 이 과정을 도시한다.

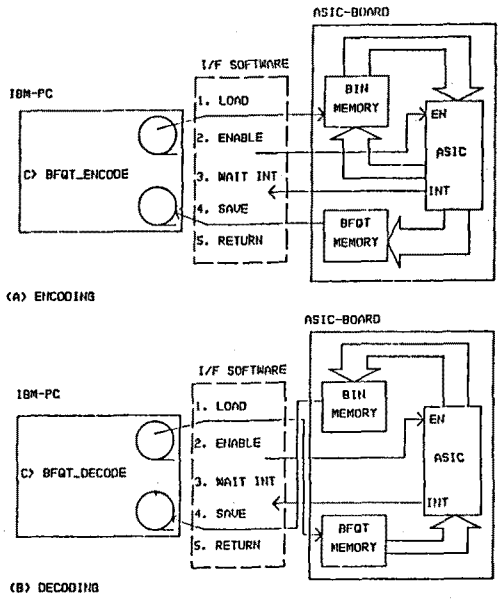


그림 2. ASIC 보드의 시스템 인터페이스

IV. ASIC의 개략적인 기능 구조

ASIC의 내부 구성은 압축부와 재생부로 크게 나누어진다.

IV.1. 압축부

압축부는 이진 화상 정보를 BFQT 표현으로 만들어 주는 부분 으로서 다음과 같은 모듈들, 즉 주소부, 처리부, 축소부, BFQT 생성부로 구성된다.

압축부는 이진 화상이 보관된 BIN MEMORY로부터 화상 정보를 가져 오는 과정이 필요하므로 BIN MEMORY내의 데이타를 채취하기 위한 주소 생성 회로가 필요하며 이 부분을 주소부라 부른다. 이 주소부에서 생성된 주소를 이용하여 이진 화상 정보를 4개 읽어들인 후 그 값들을 비교하여 모두 2^{level}이라면 더 이상 압축 가능하므로 축소부로 보내고, 그렇지 않다면 BFQT 생성부로 보내어 BFQT 표현을 생성하도록 해 주는 처리부가 필요하다. 처리부의 수행 결과에 따라서 더 이상 압축할 수 있으면 축소부로 보내어 화상 데이타를 압축시키서 다음 레벨에서의 처리를 도와 주며, 그렇지 않으면

BFQT 생성부로 보내어 이진 화상 정보를 BFQT 표현으로 바꾸어서 BFQT MEMORY에 저장한다.

IV.2. 재생부

재생부는 BFQT 표현을 본래의 이진 화상으로 복구시켜 주는 부분 으로서 다음과 같은 모듈들, 즉 레벨부, 주소부, 처리부로 구성된다.

재생부는 BFQT MEMORY내의 BFQT 표현의 레벨수로부터 레벨값을 계산 해야 하므로 레벨부를 두었다. 또한 이 레벨부로부터 인에이블 신호를 받아서 BFQT MEMORY로부터 BFQT 표현을 읽어들이는 데 필요한 주소를 생성 해 주는 주소부가 필요하다. 이제 주소부에서 생성된 주소를 이용하여 BFQT 표현을 이진 화상 정보로 바꾸어서 BIN MEMORY에 저장해 주는 기능을 수행하기 위하여 처리부를 필요로 한다. 그림 3에 ASIC의 내부 구조를 도시한다.

V. 하드웨어 알고리즘

화상 처리 ASIC 설계에 있어서 ASIC의 개략적인 기능 구조와 관련지어 BFQT 알고리즘의 하드웨어화를 위하여 레지스터 수준의 논리 회로 설계로 변환 가능한 하드웨어 알고리즘을 기술한다.

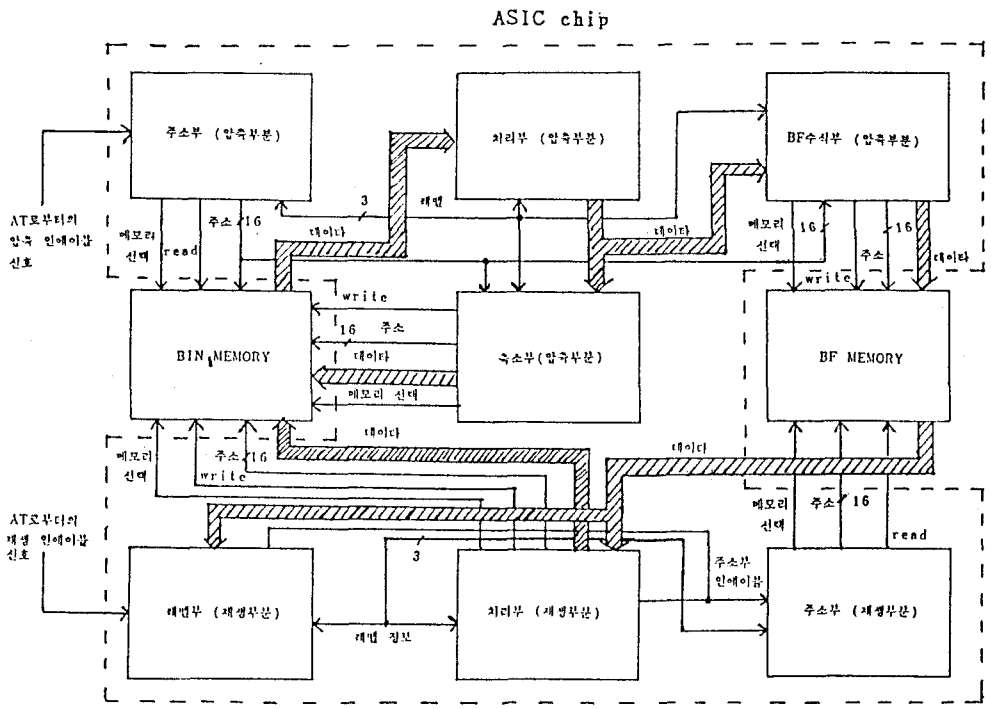
V.1. ASIC 압축부 하드웨어 알고리즘

1. IBM PC가 ASIC으로 압축부 인에이블 신호를 줌으로서 ASIC은 동작을 개시한다.

2. 주소부내의 레벨 증가기를 clear시킨다.

3. 각 레벨에서 주소를 초기화시킨다.

4. 주소부에서는 처리할 화상 데이타를 fetch하기 위하여 BIN MEMORY의 주소를 생성하고, 그 주소를 BFQT 생성부로 전송한다. 주소부에서 생성된 주소로써 BIN MEMORY의 내용을 읽어서 처리부의 버퍼에 저장한다. 이 과정을 4회 반복한다. 4개의 주소 가운데 첫번째로 생성된 주소만 축소부로 전송한다.



주) 점선으로 연결된 부분이 ASIC chip으로 제작될 예정이다.

그림 3. ASIC의 내부 구조

5. /* 처리부내의 4개의 버퍼들의 level값에 대응하는 비트들을 검사하여, 그 값이 1이면 축소부로 버퍼 0의 값을 보내고, 그 값이 0이면 BFQT 생성부로 버퍼 0,1,2,3의 값을 보낸다. */

if (buffer[0][level] + buffer[1][level] + buffer[2][level] + buffer[3][level]) = 4
then transmit buffer[0] into data_buf of 축소부;
else transmit buffer[0],buffer[1],buffer[2],and buffer[3] into data_buf[0..3] of BFQT 생성부;

6. BFQT 생성부는 데이타 버퍼들의 값을 검사하여 0이 아니면 그에 대응하는 주소 버퍼들의 값을 BFQT MEMORY에 저장한다.

7. 축소부는 처리부에서 전송된 데이타를 1비트 좌향 자리 이동한 값을 주소부에서 전송된 주소를 이용하여 BIN MEMORY에 저장한다.

8. write data_buf on BIN MEMORY with addr_buf of 축소부;

9. 각 레벨의 마지막 주소를 계산하여 그 주소의 내용을 처리했는 지를 검사한다. 처리했다면, level값을 1 증가시킨다. 그렇지 않다면, step 4로 분기하여 계속 수행한다.

final address = ba + 256 * index + offset

이때, index = offset = $256 - 2^{level+1}$

10. level값이 8이 되면 ASIC의 압축 과정이 종료되고 제어가 IBM PC로 넘어간다. 그렇지 않으면 step 3으로 분기하여 다음 level부터 계속 수행된다.

V.2. ASIC 재생부 하드웨어 알고리즘

1. IBM PC가 ASIC으로 재생부 인에이블 신호를 줌으로서 ASIC은 동작을 개시한다.

2. 레벨부는 BFQT MEMORY에서 레벨수를 읽어들인다.

3. /* 레벨값을 계산하여 초기화 한다. */

level <-- 8 - (number of levels)

4. 레벨부는 주소부 인에이블 신호를 주소부로 보낸다.

5. /* 주소부는 해당 레벨의 검은 노드의 수를 읽어서 처리부로 전송 */

load number of black nodes into counter_value of 처리부;

6. /* 처리부는 해당 레벨에서의 검은 노드들의 BFQT 표현을 본래의 이진 화상 표현으로 바꾸어서 BIN MEMORY에 저장한다. */

while counter_value > 0 do
begin
/* 처리부는 BFQT MEMORY의 데이타부의 내용을 읽어들인다. */
read BFQT 표현 from 데이타부 of BFQT MEMORY;
/* BFQT MEMORY의 데이타부의 내용은 압축된 BIN MEMORY의 가상주소이므로 이 값을 base address로 사용하여 본래의 이진화상으로 변환한다. */
base address(ba) <-- virtual address(va) from 데이타부;
for row = 0 to $2^{7-level} - 1$ do
for col = 0 to $2^{7-level} - 1$ do
begin
/* 본래의 이진 화상의 virtual address(va)를 계산한다. */
va <-- ba + 256 * row + col;
/* 가상주소를 BIN MEMORY의 실제주소로 변환한다. */
convert va to pa using routing technique;
/* 실제주소로써 BIN MEMORY상에 1을 기록한다. */
write '1' on BIN MEMORY with pa;
end; /* for loop */
/* 나머지 검은 노드들도 마찬가지로 처리한다. */
counter_value <-- counter_value - 1;
end; /* while loop */

7. /* 현재 레벨의 검은 노드를 모두 처리한 뒤에

는 다음 레벨로 신행한다. */

level <-- level + 1

8. /* level값이 8이 되면 ASIC의 재생 과정이 종료되고 제어가 IBM PC로 넘어간다. 그렇지 않으면 step 5로 분기하여 다음 level에서 부터 계속 수행 된다. */

VI. 결과 하드웨어

앞 장에서 기술된 하드웨어 알고리즘을 이용하여 레지스터 수준의 논리 회로를 설계한다[6]. 이 장에서는 ASIC의 논리 회로 부분을 모두 나타낼 수는 없으므로 압축부내의 처리부와 재생부내의 처리부만을 명시한다.

그림 4에 ASIC 압축부 하드웨어 알고리즘 중에서 과정 5에 해당하는 처리부를 보인다.

그림 5에 ASIC 재생부 하드웨어 알고리즘 중에서 과정 6에 해당하는 처리부를 보인다.

VII. 성능 평가

VII.1. 압축 시간

데이타(참고 문헌 [7] 참조)를 가지고 IBM PC/AT (클럭 = 6MHz)상에서 BFQT 소프트웨어[3]로 압축했을 때의 시간은 약 15초가 걸렸다. 이를 하드웨어로 수행했을 때의 예상 시간은 약 0.7초가 걸린다. 이것은 다음과 같은 수식으로 추정할 수 있다. IBM PC/AT의 클럭을 6MHz로 가정했을 때의 1클럭 주기는 약 167ns이다. 메모리에서 데이타를 읽거나 쓸때는 일반적으로 6클럭 정도가 소요된다[5].

앞에서도 설명한 바와 같이 인터페이싱 명령(BFQT_ENCODE)에 의해서 IBM PC/AT의 프레임 버퍼에서 BIN MEMORY로 옮겨진다. 하드웨어의 수행은 이 BIN MEMORY에서 4바이트를 fetch해서 처리하여 압축된 결과를 저장 또는 BFQT 표현으로 BFQT MEMORY에

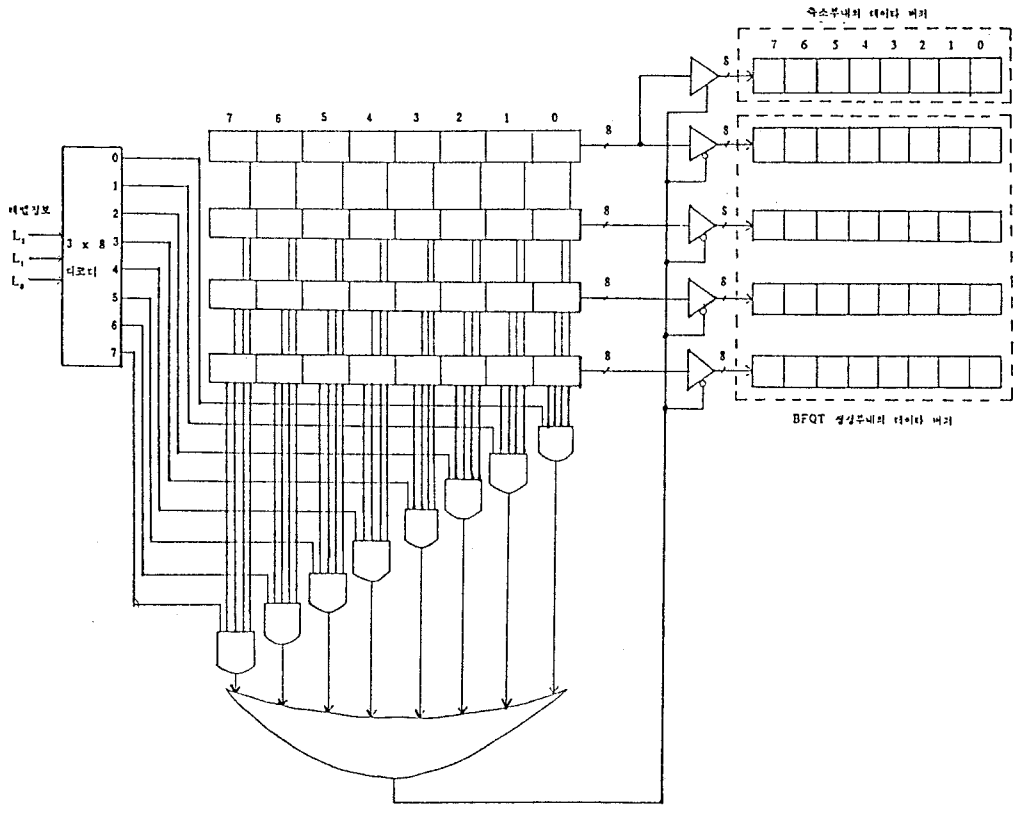


그림 4. 압축부내의 처리부

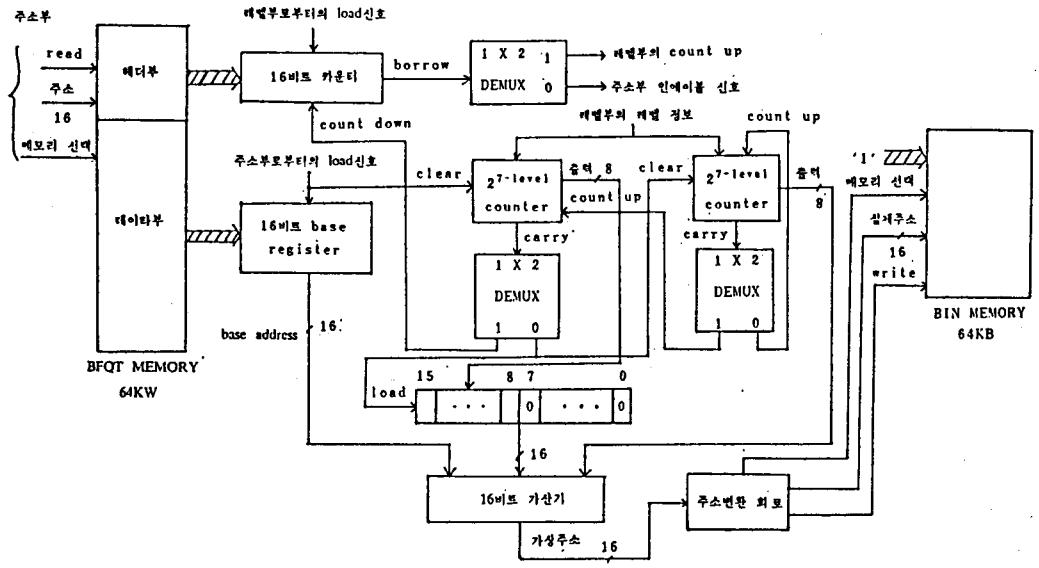


그림 5. 재생부내의 처리부

저장하는 일련의 과정을 한 cycle로 볼때 한 cycle의 경과 시간, 즉 cycle_time은

cycle_time = 4 번의fetch + 처리(압축) + 1 번의 저장 = 약 6(μs)이된다.

또한 레벨 i(i=1,2,..,8)에서의 cycle수,즉 level_cycle(i)은

level_cycle(i) = 64K / 4^i이다.

따라서 전체 레벨의 cycle수에 cycle_time을 곱한 값이 전체 레벨의 인코딩 시간이 된다.

전체 레벨 인코딩 시간 $= \sum_{i=1}^{8}$ level_cycle(i) * cycle time + α = 약 200(ms). 여기서 α 는 여유값임.

한편 인터페이싱 시간은 64K의 이진 화상 화일로부터 읽어서 BIN MEMORY에 기록하고, 또 BFQT MEMORY로부터 읽어서 BFQT 표현 화일로 기록하므로 추정시간은 다음과 같다.

인터페이싱 시간 = (64K fetch + 64K store) + β = 약 500(ms)

여기서 β는 BFQT MEMORY에서 BFQT 표현 화일로 만드는데 필요한 시간으로 약 300(ms) 이다.

따라서 총 하드웨어 압축 시간은 다음과 같다.

총 하드웨어 압축시간 = 전체 레벨 인코딩 시간 + 인테페이싱 시간 = 약 700 (ms)

그리므로 소프트웨어로 처리한 시간 보다 약 21배 정도가 빠르게 추정된다.

VII.2. 재생 시간

같은 데이타[7]로 IBM PC/AT상에서 BFQT 소프트웨어로 재생시켰을 때의 시간은 약 3초가 걸린다. ASIC으로 재생할 때 최악의 경우를 고려하면, 64K의 BFQT 화일을 읽어 64K의 이진 화상을 만들어 내는 경우이다.

그러므로 최악의 디코딩 시간은

최악 디코딩 시간 = 64K fetch + 64K store + σ = 약 200(ms)이다. 여기서 σ는 여유값 임.

따라서 총 하드웨어 재생 시간은

최악 디코딩 시간 + 인터페이싱 시간 = 약 700 (ms)이다.

그리므로 소프트웨어로 처리한 시간 보다 약 4배 정도 빠르게 추정된다.

VIII. 결론

본 논문은 화상처리 알고리즘 중에서 기본적이면서 많은 시간이 소요되는 압축과 재생부분을 하드웨어로 구현했다. 인간에게 많은 정보를 제공해 주는 화상을 컴퓨터로 처리하기 위해서는 많은 기억 장소와 시간이 소요되므로 이를 하드웨어로 구현함으로써 속도가 향상되어 X-ray등 실시간이 요구되는 분야에 이용 가능해진다.

본 논문에서 제안한 하드웨어 알고리즘은 소프트웨이를 하드웨어로 구현하려는 다른 시도에도 유용하리라고 본다. 한편 하드웨어내의 timing을 고려한 제어신호에 대한 좀더 명확한 제시가 요구됨에 따라 이에 대한 연구를 계속 발전시켜 나갈 것이다.

참고 문헌

[1] ASIC의 개발동향과 효과적인 활용법, 전자재료, 1986

[2] 이 민규,김 민환,황 희웅,"2진 영상의 효과적인 표현법:BF Linear Quadtree," 대한 전기 학회 전산기 연구회,추계 학술 연구 발표회 논문집,pp.23-28,1985

[3] 서울대학교,화상의 구조적 표현에 관한 연구,한국 재단 최종보고서,1987

[4] 소림방직, ASIC의 논리회로 설계법,1988

[5] Lewis C. Eggebrecht; Interfacing to the IBM Personal Computer,1983

[6] M.Morris Mano, Digital Design, 1984

[7] 이 민규, "이진 화상의 효율적인 표현 방법 (BF Gray Quadtree)과 연산에 관한 연구," 서울 대학교 전자계산기공학과 석사 학위 논문,1988