고속 프랙탈 영상압축을 위한 VLSI 어레이 구조

정회원 성 길 영*, 이 수 진**, 우 종 호***

VLSI Array Architecture for High Speed Fractal Image Compression

Kil-Young Sung* , Su-Jin Lee** , Chong-Ho Woo*** *Regular Members*

요 약

본 논문에서는 쿼드-트리 방식을 이용한 프랙탈 영상압축 알고리즘의 고속화를 위한 1-차원 VLSI 어레이를 제안한다. 먼저, 순차적 Fisher 알고리즘을 단일할당코드 알고리즘으로 변환하여 데이터의존 그래프를 구하였다. 구해진 데이터의존 그래프를 최적의 방향으로 투영시켜 2-차원 어레이를 설계하고, 구해진 2-차원 어레이를 변형하여 1-차원 VLSI 어레이를 설계하였다. 설계한 1-차원 VLSI 어레이에서 치역블록 및 정의역블록을 입력하는 핀과 처리요소의 내부 연산장치를 공유함으로써 입출력 핀의 수를 줄이고 처리요소의 구조를 간단하게 했다. 또한 각 블록크기에 대한 연산을 위한 처리요소를 재사용하여 처리요소의 이용률을 높였다. 512×512 그레이-스케일 영상의 프랙탈 영상압축에 대하여, 제안한 어레이는 순차적인 알고리즘에 비하여 약 67배 이상 빠르게 수행될 수 있다. 제안한 1-차원 VLSI 어레이의 동작은 컴퓨터 시뮬레이션을 통하여 검증하였다.

ABSTRACT

In this paper, an one-dimensional VLSI array for high speed processing of fractal image compression algorithm based the quad-tree partitioning method is proposed. First of all, the single assignment code algorithm is derived from the sequential Fisher's algorithm, and then the data dependence graph(DG) is obtained. The two-dimensional array is designed by projecting this DG along the optimal direction and the one-dimensional VLSI array is designed by transforming the obtained two-dimensional array. The number of Input/Output pins in the designed one-dimensional array can be reduced and the architecture of process elements(PEs) can be simplified by sharing the input pins of range and domain blocks and internal arithmetic units of PEs. Also, the utilization of PEs can be increased by reusing PEs for operations to the each block-size. For fractal image compression of 512×512 gray-scale image, the proposed array can be processed fastly about 67 times more than sequential algorithm. The operations of the proposed one-dimensional VLSI array are verified by the computer simulation.

Ⅰ. 서 론

최근 디지털 영상의 활용이 증가함에 따라 DCT, 웨이브렛, JPEG, 프랙탈[1-4] 등을 이용한 영상압축에 대한 연구가 활발히 진행되고 있다. 프랙탈 영상압축은 코드북(codebook)이 필요 없으며 고압축비, 고속도 복원, 해상도 독립성과 다른 기술들과의 조합 등에서 우수한 특징을 나타내고 있다[5].

프랙탈 압축 알고리즘은 많은 계산량으로 인하여 부호화에 많은 시간이 걸리는 단점을 갖는다[6]. 그러나 알고리즘의 병렬성이 매우 높으므로 대량의

* 경상대학교 정보통신공학과, 해양산업연구소
** 부경대학교 전자공학과 *** 부경대학교 컴퓨터멀티미디어공학부
논문번호 : 00036-0127, 접수일자 : 2000년 1월 27일

데이터를 동시에 처리하는 VLSI 어레이를 구현하여 처리속도를 크게 향상시킬 수 있다.

1995년 Y. Fisher는 정의역블록 분류의 최적화를 사용하는 순차 머신의 쿼드-트리 분할 알고리즘을 이용하여 실행시간, 압축비, 그리고 첨두 신호 대 잡음비(PSNR) 등을 절충하는 방법을 제시하였다[7]. 1996년 F. Ancarani는 PCI 버스를 이용하여 PC 플랫폼의 프랙탈 압축을 위한 병렬 주문형 집적회로(ASIC) 구조를 제안하였다. 이 ASIC을 이용하면 순차적인 알고리즘보다 약 300배의 속도향상을 얻을 수 있다. 그러나 고정블록분할 방법을 이용하여 압축비가 낮으며, 치역블록과 인접한 정의역만을 비교하므로 신뢰도가 저하된다[8]. 1997년 D. J. Jackson은 순환 계산 모델(circulating computation model)을 사용하는 128개의 처리요소로 구성된 nCube 다중처리 시스템에 쿼드-트리 분할방식의 프랙탈 압축 알고리즘을 사상했다. 하나의 처리기가 호스트 처리기의 역할을 수행하고, 나머지 처리기들은 슬레이브 처리기의 역할을 수행한다. 이 시스템은 처리요소의 구조가 복잡하여 VLSI로 구현에 적합하지 않다[9].

본 논문에서는 쿼드-트리 분할방식의 알고리즘을 변형하여 단일할당코드로 변환하고, 이를 근거로 1-차원 시스톨릭어레이를 설계한다. 설계한 어레이는 19개의 처리요소(process element: PE)들로 구성되며, 141개의 데이터 입출력 핀을 갖는다. 4×4 크기의 치역블록에 대해 정합하는 정의역블록을 찾기 위한 PE들로 어레이를 구성하며, 반복수행을 통해 8×8 및 16×16 크기의 치역블록에 대한 정합하는 정의역블록을 찾는다. 먼저 16×16 크기의 치역블록에 대해 정합하는 정의역블록의 좌표와 s(scaling coefficient) 및 o(offset coefficient)의 값을 호스트 컴퓨터에 전송하고, mse(mean square error)가 임계값을 만족하지 않는 치역블록들에 대해 8×8 크기의 블록으로 분할하여 정합되는 정의블록을 찾는다. 같은 방법으로 4×4 크기의 치역블록에 대한 비교연산을 수행한다. 이 시스톨릭어레이를 이용하면 높은 신뢰도를 유지하면서 순차적인 알고리즘에 비해 속도가 약 67배로 향상된다.

Ⅱ. 알고리즘의 병렬화와 데이터의존 그래프

프랙탈 영상압축은 영상을 치역 및 정의역블록으로 나누고 자기유사성을 찾아 데이터의 양을 줄인다. 그림 1은 Fisher의 고정블록 분할방법의 프랙탈 영상압축 알고리즘이다. 일반적인 쿼드-트리 분할방식의 알고리즘은 데이터의 입출력이 불규칙적이므로 시스톨릭어레이로 구현하기가 적합하지 않다. 본 논문에서는 데이터 입력이 규칙적인 고정분할 방식 알고리즘을 치역블록의 크기에 따라 반복수행하여 쿼드-트리 분할방식을 적용한 효과를 얻는다.

```
for m=0 to (M-1)    ; M : the number of range
    for l=0 to (L-1)  ; L : the number of domain
        s1 = s2 = s3 = s4 = 0
        for i=0 to (n-1)
            s1 = s1 + d_i · r_i
            s2 = s2 + d_i
            s3 = s3 + r_i
            s4 = s4 + d_i^2
            s5 = s5 + r_i^2
        next i
        s6 = s2^2
        s = (n s1 - s2 s3) / (n s4 - s6)
        o = (s3 - s s2) /n
        mse = (s^2 s4 + n · o^2 + s5 + 2s · o · s2
                - 2o · s3 - 2s · s1)/n
        if(mse < p_mse)
            s, o , m, l, mse 값 저장
        end if
    next l
next m
```

그림 1. 순차적 Fisher 알고리즘

Fisher의 알고리즘은 치역블록과 정의역블록 사이에 자기유사성을 탐색하기 위해 많은 수의 블록을 비교하므로 과도한 계산량이 요구된다. 그러나 이 알고리즘은 규칙적인 데이터 흐름을 가지므로 병렬 알고리즘으로의 변형이 용이하다. 그림 2는 인덱스 확장법을 사용하여 그림 1의 순차적 알고리즘을 병렬 알고리즘 표현의 한 형태인 단일할당코드 알고리즘으로 변환한 결과이다[10]. 이 알고리즘은 치역 및 정의역블록의 각 픽셀값을 연산하는 A 부분, s를 계산하는 As 부분, o를 계산하는 Ao 부분 및 치역과 정의역블록의 오차 mse를 계산하는 Amse로 구성된다.

단일할당코드 알고리즘으로부터 데이터의 의존관계를 파악하여 그림 3과 같은 3차원의 데이터 의존 그래프를 유도한다. 치역블록의 픽셀값은 [m l i]=[0

```
for m=0 to (M-1)
    for l=0 to (L-1)
        s1(m, l, 0) = 0;
        s2(m, l, 0) = 0;
        s3(m, l, 0) = 0;
        s4(m, l, 0) = 0;
        s5(m, l, 0) = 0;
        for i=0 to (n-1)
A           s1(m, l, i) = s1(m, l, i-1) + d_i · r_i
            s2(m, l, i) = s2(m, l, i-1) + d_i
            s3(m, l, i) = s3(m, l, i-1) + r_i
            s4(m, l, i) = s4(m, l, i-1) + d_i^2
            s5(m, l, i) = s5(m, l, i-1) + r_i^2
        next i
As      s(m, l) = (n*s1(m, l, n) - s2(m, l, n)
                  *s3(m, l, n))/(n*s4(m, l, n)
                  - s2(m, l, n)*s2(m, l, n))
Ao      o(m, l) = (s3(m, l, n) - s(m, l)
                  *s2(m, l, n))/n
        mse(m, l) = (s(m, l)*s(m, l)
                  *s4(m, l, n) + n*o(m, l)
                  *o(m, l) + s5(m, l, n)
                  + 2*s(m, l)*o(m, l)
                  *s2(m, l, n) - 2*o(m, l)
Amse              *s3(m, l, n) - 2*s(m, l)
                  *s1(m, l, n))/n
        if(mse(m, l) < mse(m, l-1))
            m, l, s (m, l), o(m, l), mse(m, l)
              저장
        end if
    next l
next m
```

그림 2. 단일 할당 코드 알고리즘

1 0]의 방향으로 입력되고, 정의역블록의 픽셀값은 [1 0 0] 방향으로 입력된다. 이 값들을 누적하는 A PE의 계산값은 [0 0 1] 방향으로 전송되어 As 및 Ao PE에서 s와 o값을 계산한다. Amse PE에서는 치역과 정의역블록의 자기유사성 정도인 mse를 계산하고 [0 1 0] 방향으로 전송하여 치역블록에 정합되는 정의역블록을 탐색한다. 이 데이터 의존 그래프인 그림3에는 식(1)과 같은 데이터의존벡터가 존재한다.

$$D = [\ \vec{e_1}\ \ \vec{e_2}\ \ \vec{e_3}\ ] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

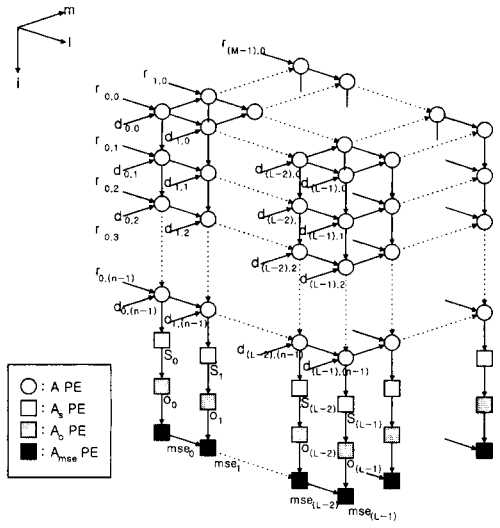

그림 3. 3차원 데이터 의존 그래프

Ⅲ. 시스톨릭어레이의 설계

시스톨릭어레이는 데이터 의존 그래프를 적당한 방향으로 투영시켜 시간 및 공간 사상을 통해 얻어진다[10]. 데이터 입출력 핀의 수를 최소로 하기 위하여 그림 3의 데이터 의존 그래프를 [0 1 0] 방향으로 투영하여 2-차원 평면의 어레이 구조를 얻는다. 알고리즘이 투영된 어레이에서 정상적인 동작을 위해서는 식(2)를 만족하여야 한다.

$$\vec{s}^T \vec{e_i} > 0 \quad , \quad \vec{s}^T \vec{d} > 0 \tag{2}$$

여기서 $\vec{d}$는 투영벡터이고, $\vec{s}^T$는 스케줄벡터이다. 식(2)를 만족하는 최적의 스케줄벡터 $\vec{s}^T$는 식(3)과 같다.

$$\vec{s}^T = [1\ 1\ 1] \tag{3}$$

공간변환을 위한 공간변환벡터 P^T는 투영벡터와 직교하므로 식(4)와 같다.

$$P^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

따라서 스케줄벡터와 공간변환으로 구성되는 변환벡터 T는 식(5)와 같다.

$$T = \begin{bmatrix} \vec{s}^T \\ P^T \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

식(1)의 데이터 의존 벡터를 *m-i* 평면으로 투영하면 식(6)과 같다.

$$\begin{aligned}\vec{e}' &= P^T \vec{e}_i \\ &= [\ \vec{e}_1'\ \ \vec{e}_2'\ \ \vec{e}_3'\ ] \\ &= \begin{bmatrix}1&0&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \\ &= \begin{bmatrix}1&0&0\\0&0&1\end{bmatrix}\end{aligned} \tag{6}$$

따라서 치역블록은 각 PE에 선적재되고, 정의역 블록은 [1 0] 방향으로 입력되며, A PE의 중간계산 값은 [0 1] 방향으로 전송된다. 2-차원 어레이로 사상된 후 각 의존벡터들의 시간지연은 식(7)과 같다.

$$\begin{aligned}D(\vec{e}') &= \vec{s}^T \vec{e}_i \\ &= [D(\vec{e}_1')\ D(\vec{e}_2')\ D(\vec{e}_3')\ ] \\ &= [1\ 1\ 1]\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \\ &= [1\ 1\ 1]\end{aligned} \tag{7}$$

그림3의 데이터의존 그래프의 각 에지들은 단위 시간의 시간지연을 가지므로 시스톨릭어레이의 시간적 지역성을 만족한다.

$\vec{d} = [0\ 1\ 0]^T$, $\vec{s}^T = [1\ 1\ 1]$ 으로 선택한 경우의 시스톨릭어레이는 그림4와 같다.

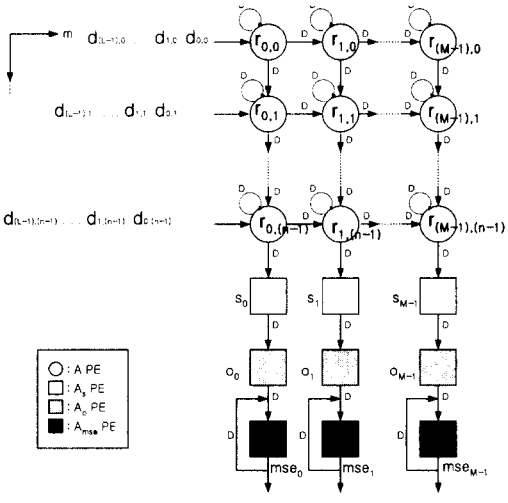

그림 4. 2차원 시스톨릭 어레이

그림4에서 원으로 표현된 A PE는 영상의 픽셀값을 입력하기 위한 8-비트의 데이터 입력핀이 존재하며, Amse PE에는 정합여부를 출력하기 위한 1-비트의 출력핀과 선택된 s를 출력하기 위한 5-비트 및 o를 출력하기 위한 7-비트의 데이터 출력핀을 갖는다.

그림 4의 2-차원 시스톨릭어레이는 영상의 크기에 따라 PE의 수가 가변적이며 PE의 수가 너무 커져서 VLSI 구현에 문제점이 있다. 또한 쿼드트리의 각 깊이에 대해 열의 길이가 다르므로 각 깊이에 대한 어레이가 각각 존재해야 한다. 이러한 문제점을 보완하기 위해 그림4의 2-차원 어레이를 [*m i*]=[1 0] 방향으로 투영하면 그림 5와 같은 1-차원 시스톨릭어레이로 변환된다.

그림 5의 1-차원 시스톨릭어레이는 정의역 및 치역의 픽셀값과 제곱의 값을 각각 누적하는 16개의 PE와 s, o, mse를 연산하고 최적의 값을 기억하는 3개의 PE로 구성된다.

어레이에서 데이터의 입력 순서는 A PE에 블록의 크기를 지정하는 n의 값이 입력된 후, 치역과 정의역블록이 차례로 입력된다.

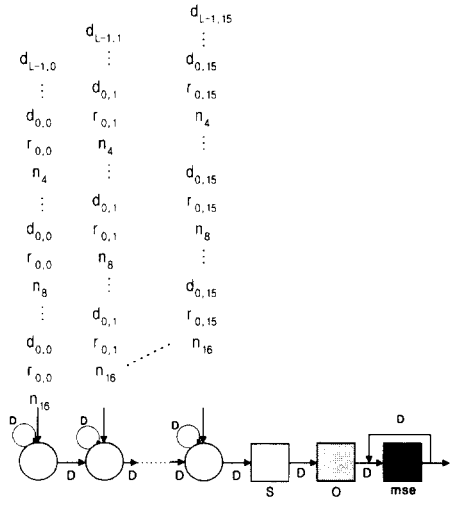

그림 5. 1차원 시스톨릭 어레이

16×16의 크기에 해당하는 치역 및 정의역 값이 교대로 입력되어 각 블록들에 대한 mse값을 구한다. 각 치역블록에 대해 정합하는 정의역블록의 좌표와 s, 및 o값을 계산하여 호스트컴퓨터에 전송한다. 임계값을 만족하지 않는 치역블록들은 다시 8×8 크기의 블록으로 정합하는 블록을 찾는다. 같은

방법으로 4×4크기의 블록에 대해서 정합하는 블록을 찾는다. 16×16과 8×8 크기의 블록에 대해서는 치역 및 정의역 값이 교대로 입력하여 계산하고, 4×4 크기의 블록에 대해서는 한번 입력된 치역을 모든 정의역에 대해서 재사용되도록 PE 내부에 저장해 둔다.

따라서 결과의 1-차원 시스톨릭어레이는 원영상의 크기에 독립적인 프랙탈 영상압축을 구현하는 VLSI 어레이 구조가 된다.

Ⅳ. 시뮬레이션과 결과 및 고찰

설계한 VLSI 어레이의 동작을 검증하기 위하여 64×64 크기의 그레이-스케일의 Lenna 영상을 이용하여 컴퓨터시뮬레이션을 수행하였다.

1. 시뮬레이션 환경

(1) 프로그래밍 언어 : VC++ 5.0
(2) 컴퓨터 : 펜티움 II 500
(3) 사용영상 : 64×64 Gray-Scale Lenna 영상

2. 시뮬레이터 동작 순서

(1) Lenna 영상을 블록의 크기에 따라 치역과 정의역 풀로 나누어 메모리에 저장한다.
(2) 16×16 크기의 치역과 정의역블록을 입력하여 정합하는 블록을 찾는다.
(3) 임계치를 만족하는 정의역블록이 없 는 치역블록은 호스트컴퓨터의 역할 을 하는 부분에서 기억한다.
(4) 모든 16×16 크기의 블록에 대해 연 산이 끝난 후 정합되지 않는 블록을 8×8크기로 분할하여 어레이에 입력 한다.
(5) 같은 방법으로 8×8과 4×4 크기의 블록들에 대해 수행한다.
(6) 호스트컴퓨터의 역할을 하는 부분에 저장된 치역 및 정합하는 정의역의 인덱스와 s, o를 파일로 저장한다.

그림 6은 16번째 클럭에서 어레이의 동작을 나타내며, 사각형은 PE를 표현한 것이며 PE 내부의 숫자는 중간계산값을 저장하기 위한 레지스터의 내용을 표현한 것이다. 먼저 16×16 크기의 치역블록에 정합하는 정의역블록을 탐색하기 위해 블록의 크기와 치역 및 정의역 픽셀값을 차례대로 입력한다. 그림 7은 4610 클럭에서 어레이의 상태를 나타낸 것으로 16×16 블록크기에 해당하는 데이터를 모두 입력한 후 mse가 임계값을 만족하지 않는 치역블록들을 8×8 크기로 분할하여 입력되는 것을 확인할 수 있다. 또한 같은 방법으로 8×8 크기의 치역블록에 정합하는 정의역블록을 탐색하고, mse를 만족하지 못하는 치역블록을 4×4 크기로 분할하여 입력한다. 4×4 크기의 블록에 대해 mse값이 최소인 정의역블록을 정합하는 블록으로 선택한다.

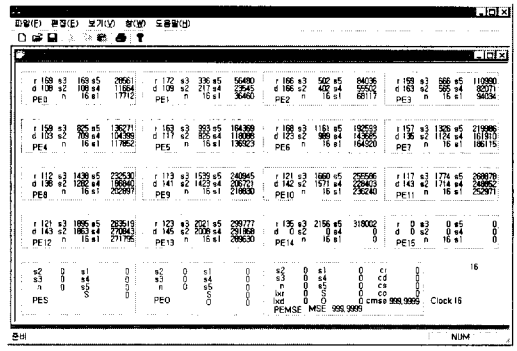

그림 6. 16 단위시간 경과 후의 1-차원어레이의 내부상태 (64×64 크기 Lenna영상)

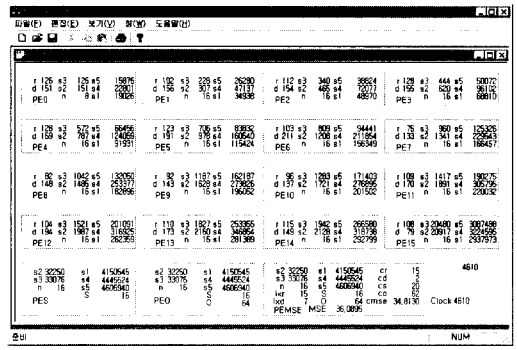

그림 7. 4610 단위시간 경과후의1-차원어레이의 내부상태 (64×64 크기 Lenna영상)

설계한 1-차원 어레이에 16×16 크기를 갖는 하나의 치역과 정의역의 픽셀값을 입력하기 위해서 16번 반복해서 입력해야한다. 또한 치역과 정의역블록의 픽셀은 순차적으로 입력해야하므로 32개의 단위시간이 필요하다. 따라서 16×16 크기를 갖는 치역과 정의역블록의 수를 각각 R16, D16이라고 하면, 16×16 크기의 모든 블록은 입력하는데 R16×D16×32 의 단위시간이 필요하다. 8×8 크기의 치역 및 정의역블록의 픽셀값을 1-차원 어레이에 입력하기 위해서 4번 반복해서 입력해야한다. 그러므로 8×8 크기의 치역 및 정의역블록의 수를 각각 R8, D8이라고 하면 블록 전체를 입력하는데 R8×D8×8

의 단위시간이 걸린다. 4×4 크기의 블록은 치역블록의 값을 선적재시키고 정의역블록들을 차례로 입력하므로 치역 및 정의역블록의 수를 각각 R4, D4라면 R4×(D4+1)의 단위시간이 필요하다. 따라서 VLSI 어레이에 블록의 크기와 모든 치역 및 정의역블록의 픽셀들이 입력되어 정합되는 치역과 정의역블록의 쌍을 구하는데 (21+R16×D16×32+R8×D8×8+R4×(D4+1))의 단위시간이 필요하다. 이 VLSI 어레이를 이용하여 512×512그레이 스케일 영상에 대한 프랙탈 영상압축은 순차적인 알고리즘에 비해 약 67배 이상 빠르게 수행된다.

설계한 프랙탈 영상압축을 위한 1-차원 VLSI 어레이는 데이터 입출력핀이 141개로 고정되고, 데이터의 입출력이 규칙적이므로 VLSI 어레이의 구현에 적합하다. 또한 원본영상의 크기에 상관없이 같은 형태를 갖는다.

V. 결 론

본 논문에서는 VLSI 어레이 구현에 용이한 프랙탈 영상압축의 고속화를 위한 1-차원 시스톨릭어레이를 설계하였다. 프랙탈 압축 알고리즘을 단일할당 코드로 변환하여, 데이터 의존관계를 구하고, 시간 및 공간 변환을 통하여 2-차원 시스톨릭어레이를 유도하였다. 유도된 2-차원 어레이로부터 PE의 개수와 입출력핀을 줄여서 VLSI 구현에 적합한 1-차원 시스톨릭어레이를 설계하였다.

블록의 크기에 대한 어레이를 독립적으로 두지 않고 4×4의 치역블록에 대한 PE들만으로 어레이를 구성하였다. 그러므로 설계한 1-차원 시스톨릭어레이는 치역 및 정의역 값을 누적 및 연산하기 위한 16개의 PE와 s, o, mse를 계산하기 위한 3개의 PE, 즉 총 19개의 PE로 구성된다. 각 PE들은 입출력 핀의 숫자를 줄이기 위해 치역블록의 크기와 치역블록 및 정의역블록의 값을 입력하는 핀을 공유한다.

설계한 어레이의 데이터 입출력 핀의 수는 141개이며, 처리시간은 최악의 경우에 (21+R16×D16×32+R8×D8×8+R4×(D4+1)) 이다. 동일한 블록의 크기를 사용한다면 PE의 구조와 PE의 개수는 영상의 크기에 독립적이다.

참 고 문 헌

[1] A. E. Jacquin, "Fractal image coding: A review," Proceedings of the IEEE, Vol. 81, pp. 1451-1465, Oct., 1993.

[2] B. Rejeb and W. Anheier, "A new approach for the speed up of fractal image coding," proc., 1997 13th Conf. DSP, Vol. 2, pp. 853-856. June, 1997.

[3] E. W. Jacobs, Y. Fisher, and R. D. Boss, "Image compression: A study of the iterated transform method," Elsevier Science Publishers B. V, Signal Processing, Vol. 29, pp. 251-263, 1992.

[4] D. M. Munro and F. Dudbridge, "Fractal block coding of images, " Electron, Lett., Vol. 28, pp. 1053-1055, 1992.

[5] K. P. Aiken, M. J. Irwin, and R. M. Owens, "A parallel ASIC architecture for efficient fractal image coding," Journal of VLSI Signal Processing 19, pp. 97-113, 1998.

[6] R. F. Uys, "Parallel implementation of fractal image compression," IEEE, Proceedings of the 1998 South African Symp. Comm. & Sig. Proc., pp. 143-148, 1998.

[7] Y. Fisher, "Fractal image compression: Theory and application," Springer-Verlag, Berlin, 1995.

[8] F. Ancarani, A. De Gloria, M. Olivieri, and C. Stazzone, "Design of an ASIC architecture for high speed fractal image compression," Proc. 9th annual IEEE International ASIC Conference and Exhibit, pp. 223-226, 1996.

[9] D. J. Jackson and T. Blom, Fractal image compression using a circulating pipeline computation model," Proceedings of the ISCA 10th International Conference on Parallel and Distributed Comp. Syst., pp. 141-144, 1997.

[10] S. Y. Kung, "VLSI Array Processors," Prentice Hall, Englewood Cliffs, NJ, 1988.

성 길 영(Kil Young Sung) 정회원

1980년 2월 : 경북대학교 전자공학과 졸업(학사)

1985년 2월 : 건국대학교 대학원 전자공학과(석사)

1998년 : 부경대학교 대학원 전자공학과 박사과정 수료

1995년~현재 : 경상대학교 정보통신공학과 교수, 해양산업연구소 연구원

<주관심 분야> 컴퓨터구조, 영상압축, 병렬처리

이 수 진(Su Jin Lee) 정회원

1995년 2월 : 부경대학교 전자공학과 졸업(학사)
1997년 2월 : 부경대학교 대학원 전자공학과(석사)
1999년 : 부경대학교 대학원 전자공학과 박사과정 수료
<주관심 분야> 병렬처리, 컴퓨터구조, 영상압축

우 종 호(Chong Ho Woo) 정회원

1978년 2월 : 경북대학교 전자공학과 컴퓨터공학전공 (학사).
1981년 2월 : 경북대학교 대학원 전자공학과 전산공학 전공(석사)
1990년 2월 : 경북대학교 대학원 전자공학과 전산공학 전공(박사).
1981년 3월~현재 : 부경대학교 컴퓨터멀티미디어공학부 교수.
1987년 8월~1988년 8월 : 미국 렌셀러공대(RPI) 전산학과 객원교수.
<주관심 분야> VLSI 어레이 알고리즘, 병렬처리, 분산멀티미디어