論文2002-39SC-3-5

데이터 재사용에 의한 고속 프랙탈 영상압축을 위한 시스토릭 어레이의 설계

(Design of Systolic Array for High-Speed Fractal Image Compression by Data Reusing)

禹 鍾 鎬 *, 李 熙 珍 *, 李 守 鎭 *, 成 吉 永 **

(Chong-ho Woo, Hee-jin Lee, Su-jin Lee, and Gil-young Sung)

요 약

프랙탈 영상압축의 고속처리를 위한 일차원 VLSI 어레이를 설계하였다. 기존의 제안된 일차원 VLSI 어레이에서 중첩되는 이웃의 정의역블럭의 데이터들을 재사용하므로서 전체 연산에 필요한 데이터의 총입력 횟수를 감소시키고, 이로 인한 전체 처리시간을 줄였다. 어레이로 입력되는 데이터의존관계를 고려하여, 입력순서가 적절히 조정되었으며, 이에 따라 처리요소들을 설계하였다. 몇몇 처리요소에는 데이터의 저장 및 경로설정을 위한 레지스터와 멀티플렉서들이 추가되었다. 따라서 영상의 크기가 N이고 블럭의 크기가 B인 경우, 이 설계는 적은 하드웨어를 추가하여 기존의 어레이보다 처리속도가 (N-4B)/4(N-B)배 향상되었다.

Abstract

An one-dimensional VLSI array for high speed processing of Fractal image compression was designed. Using again the overlapped input data of adjacent domain blocks in the existing one-dimensional VLSI array, we can save the number of total input for the operations, and so we can save the total computation time. In the design procedure, we considered the data dependences between the input data, reordered the input data to the array, and designed the processing elements. Registers and multiplexors are added for the storing and routing of the input data in some processing elements. Consequently as adding a little hardware, this design shows (N-4B)/4(N-B) times of speed-up compared with the existing array, where N is image size and B is block size.

Keyword : 프랙탈영상압축, 시간공간사상, 시스토릭 어레이, VLSI 어레이

I. 서 론

영상데이터는 데이터의 용량이 크므로 효율적인 저장과 전송을 위해서 적절한 압축이 필요하다. 프랙탈 영상압축은 영상을 구성하는 각 부분의 자기 유사성을 이용하여 데이터를 압축하는 방법으로 코드북이 필요없고, 압축률이 높고 복호화 속도가 빠르며, 해상도 독립성을 가지며 다른 영상의 부호화 기술과의 조합 등의 우수한 특성을 갖는다. 그러나 이 압축방법은 부호화과정에서 정의역블럭과 치역블럭간 과도한 비교 및

* 正會員, 釜慶大學校 電子컴퓨터情報通信工學部
(Division of Electronics Computer and Communication Eng., Pukyong National University)
** 正會員, 慶尙大學校 情報通信工學科
(Dept. of Information and Communication Eng., Kyungsang National University)

※ 본 논문은 2000년 부경대학교 학술진흥재단의 지원에 의해 수행되었습니다.
接受日字:2001年8月16日, 수정완료일:2002年1月16日

계산을 요구하는 단점을 갖는다.[1-4] 1995년 Y. Fisher는 쿼드-트리 분할 알고리즘을 제안하였고[5], 최근에는 웨이브릿 기반의 프랙탈 영상압축에서 압축률과 PSNR이 높은 결과를 얻고 있다.[10-12]

프랙탈 영상압축 알고리즘은 각 블럭들에 대한 정합의 정도를 계산하는 연산과정에서 데이터 흐름이 규칙적이고 병렬성이 높으므로, 이러한 특성을 이용하여 대량의 계산을 고속 병렬로 처리할 수 있는 방안들이 제안되어 왔다. 1996년 F. Ancarani는 고정분할 방식을 이용한 병렬 ASIC 구조를 제안하였다.[6] 1997년 K. P. Acken은 쿼드-트리 분할을 이용한 ASIC 구조를 제안하였고[7], D. J. Jackson은 n-Cube에 이를 사상하였다.[8] 2000년 성 등은 고정분할 방식을 변형하여 쿼드-트리 분할의 효과를 얻을 수 있는 VLSI 어레이를 제안했다.[13]

본 논문에서는 성 등이 제안한 기존의 VLSI 어레이에서 인접한 두 정의역블럭의 중첩되는 데이터를 재사용할 수 있는 일차원 시스토릭 어레이를 설계하였다. 데이터 재사용을 위해서 일부 처리요소에 레지스터가 추가되고, 계산값을 누적하는 마지막 처리요소에서는 레지스터 및 멀티플렉서를 추가하였다. 이러한 약간의 하드웨어의 추가로 인하여 데이터의 재입력에 소요되는 시간을 처리시간에서 배제할 수 있어서, 전체 처리속도가 최대 (N-4B)/4(N-B)배 향상되었다. 컴퓨터 시뮬레이션을 통해서 설계한 어레이의 정확한 동작을 검증하였다.

Ⅱ. 프랙탈 영상압축 및 VLSI 어레이

1. Fisher의 프랙탈 영상압축 알고리즘

프랙탈 영상압축을 위한 Fisher 알고리즘은 PIFS (Partitioned Iterated Function System)를 이용하여 영상을 부호화한다. 원영상을 치역블럭과 정의역블럭으로 나누고, 각 치역블럭에 대해 자기 유사성이 가장 큰 정합된 정의역블럭을 찾기위해 전체 영역을 탐색한다. 치역블럭은 프랙탈 코드를 생성하는 과정에서 기준이 되는 부분으로 전체 영상을 커버하며 중복되지 않는다. 정의역블럭은 일반적으로 50% 겹친 형태로 치역블럭의 크기보다 (2×2)배 크기로 우선 분할한 후 이를 평균하여 치역블럭의 크기로 축소한다. 치역블럭은 축소된 정의역블럭 모두를 비교하여 가장 가까운 고정점을 갖는 반복변환시스템의 요소를 찾아낸다. 각각의 치역블럭에 대한 저장요소는 정의역블럭의 위치 및 빛의 강도 정보(s 및 o)이다. s는 대조(contrast), o는 밝기(brightness)를 나타낸다. 블럭간의 자기 유사성 정도를 비교하기 위해 식(1)과 같은 MSE(Mean Square Error)을 사용한다. 즉, 축소된 정의역블럭의 픽셀값 (a_1, a_2, …, a_n)과 치역블럭의 픽셀값 (b_1, b_2, …, b_n)간의 MSE가 최소가 되는 블럭과 s와 o를 구한다.

$$MSE = \sum_{i=1}^{n} (s \cdot a_i + o - b_i)^2 \tag{1}$$

MSE가 최소가 되는 s 와 o를 구하기 위해서 식(1)을 s와 o에 대해서 편미분하여 0으로 두면 각각 식(2), 식(3)과 같다.

$$s = \frac{n\sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} a_i \sum_{i=1}^{n} b_i}{n\sum_{i=1}^{n} a_i^2 - (\sum_{i=1}^{n} a_i)^2} \tag{2}$$

$$o = \frac{\sum_{i=1}^{n} b_i - s\sum_{i=1}^{n} a_i}{n} \tag{3}$$

식(2)과 식(3)을 식(1)에 대입하면, MSE는 식(4)와 같다.[2]

$$MSE = \frac{1}{n}\Big\{ \sum_{i=1}^{n} b_i^2 + s(s\sum_{i=1}^{n} a_i^2 - 2\sum_{i=1}^{n} a_i R_i + 2o\sum_{i=1}^{n} a_i) + o(on - 2\sum_{i=1}^{n} b_i)\Big\} \tag{4}$$

위의 변환과정을 각 치역블럭마다 모든 정의역블럭에 대해 수행한 다음 MSE가 최소인 정의역 및 치역블럭의 좌표와 아핀변환 함수 s및 o를 찾음으로써 부호화가 이루어진다.

프랙탈 영상압축을 위한 영상의 분할 방법에서 쿼드트리분할방식은 복원영상의 화질을 유지하면서 높은 압축률을 갖는 장점이 있다. 그러나 쿼드트리분할방식은 데이터의 입출력이 불규칙적이어서 VLSI어레이의 구현에는 부적합하다. 고정분할방식은 데이터의 입출력은 규칙적이나 복원영상의 화질과 데이터 압축률을 동시에 만족하지 못한다. 성 등은 고정분할방식의 알고리즘을 블럭크기 N이 16×16, 8×8, 4×4에 대해 반복적

용하여 데이터의 입출력이 규칙적이면서 쿼드트리분할 방식의 효과를 얻는 방법을 제안했다. 프랙탈 영상압축을 위한 부호화 과정을 기술하는 고정분할방식의 Fisher 알고리즘은 그림 1과 같다.[13]

```
/* M is number of range blocks */
for m=1 to M
   smse = ∞
   /* L is number of domain blocks */
   for l=1 to L
      s1 = s2 = s3 = s4 = s5 = 0
      /* n is number of pixels in blocks */
      for i=1 to n
         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 · s2 − 2o · s2 − 2s · s1)/n
      if ( mse < smse )
         smse = mse
         select m, l, s, o, mse
      end if
   next l
next m
```

그림 1. 프랙탈 영상압축을 위한 순차적 Fisher 알고리즘

Fig. 1. Sequential Fisher algorithm for fractal image compression.

2. 알고리즘의 병렬화, 시간·공간사상 및 시스토릭 어레이

순차 알고리즘을 병렬 알고리즘으로 표현하기 위해서 우선 인덱스 확장을 통해 그림 2와 같은 단일할당코드로 변환한다.[9, 13] 이것은 치역 및 정의역블럭의 각 픽셀값을 비교 계산하는 PE_a부분, s를 계산하는 PE_s부분, o를 계산하는 PE_o부분, 그리고 MSE를 계산하는 PE_{mse}부분으로 구성된다. 이 단일할당코드로부터 인덱스 (m, l, i)의 3차원 공간에서의 데이터의존 그래프로 유도할 수 있다. 치역블럭의 픽셀값은 $[m\ l\ i]$=[0 1 0]의 방향으로 입력되고 정의역블럭의 픽셀값은 [1 0 0]의 방향으로 입력된다. PE_a의 계산값은 [0 0 1] 방향으로 전송되어 마지막 PE_a에 누적된다. PEmse에서 자기유사성 정도인 MSE를 계산하고, [0 1 0] 방향으로 전송하여 치역블럭에 정합하는 정의역블럭을 탐색한다.

```
for m=1 to M
   for l=1 to L
      s1(m,l,0) = s2(m,l,0) = s3(m,l,0) = 0
      s5(m,l,0) = s6(m,l,0) = 0
      for i=1 to n
         r(m,l,i) = r(m,l−1,i)
         d(m,l,i) = d(m−1,l,i)
         s1(m,l,i) = s1(m,l,i−1)
                   + d(m,l,i) · r(m,l,i)
PE_a     s2(m,l,i) = s2(m,l,i−1) + d(m,l,i)
         s3(m,l,i) = s3(m,l,i−1) + r(m,l,i)
         s4(m,l,i) = s4(m,l,i−1) + d(m,l,i)^2
         s5(m,l,i) = s5(m,l,i−1) + r(m,l,i)^2
      next i
PE_s  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))
PE_o  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)
PE_mse          − 2×o(m,l)×s3(m,l,n)
                − 2×s(m,l)×s1(m,l,n))/n
      mse(m,l) = min(mse(m,l), mse(m,l−1))
      select m, l, s(m,l), o(m,l)
   next l
next m
```

그림 2. Fisher 알고리즘에 대한 단일할당코드 형태의 알고리즘

Fig. 2. Single assignment code for Fisher algorithm.

신호흐름 그래프는 알고리즘의 인덱스 공간에서 데이터의존 그래프를 최적의 방향으로 투영시켜 시간공간사상을 통하여 구한다. 즉 $(m\ l\ i)$축의 투영벡터 $\vec{d}$= [1 0 0] ^{T}방향으로 투영하여 $(l\ i)$축의 신호흐름 그래프를 구하고, 이로부터 이차원 시스토릭 어레이를 유도한다. 이 때 스케줄벡터 $\vec{s}^T$ = [1 1 1]로 설정하였다.

VLSI 구현을 위하여 이 어레이를 $[l\ i]$=[1 0] 방향으로 투영하고, 각 PE_a의 입력핀을 공유하도록 일차원 어레이로 변형할 수 있다. s, o 및 MSE를 연산하고, s, o를 양자화하는 과정은 앞의 다른 계산보다 복잡하므로 각각 모두 3개의 처리요소로 분할하여 각 처리요소에서의 소요시간을 균등화하였다. 이로부터 유도한 일차원 시스토릭 어레이는 그림 3과 같다.[5, 13] PE0~PE15의 입력핀을 통해 치역과 정의역블럭의 데이터를 입력한다. 이 구조는 4×4 블럭크기에 해당하는 어레이

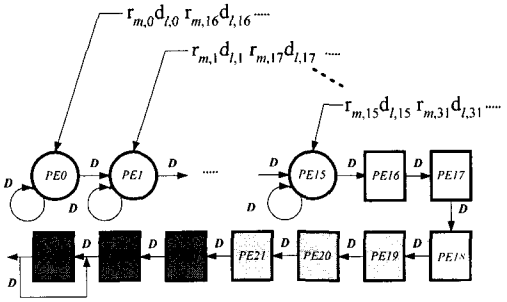

그림 3. 프랙탈 영상압축을 위한 일차원 어레이 및 입력 데이터

Fig. 3. One dimensional array and input data for fractal image compression.

구조이다. 따라서 16×16과 8×8 블럭크기 일 때는 치역과 정의역블럭을 구성하는 데이터를 반복입력하여 처리한다.

Ⅲ. 데이터 재사용에 의한 어레이의 속도개선

1. 데이터 재사용을 위한 일차원 시스토릭 어레이 설계

원영상에서 치역블럭은 B×B 크기로 분할하고, 정의역블럭은 2B×2B 크기로 인접한 정의역블럭과 50% 중첩시켜 분할한다. 2B×2B 크기의 정의역블럭은 이웃한 4개의 픽셀을 각각 평균하여 B×B크기로 축소변환한다. 그림 4는 원영상으로부터 정의역블럭들을 구성하는

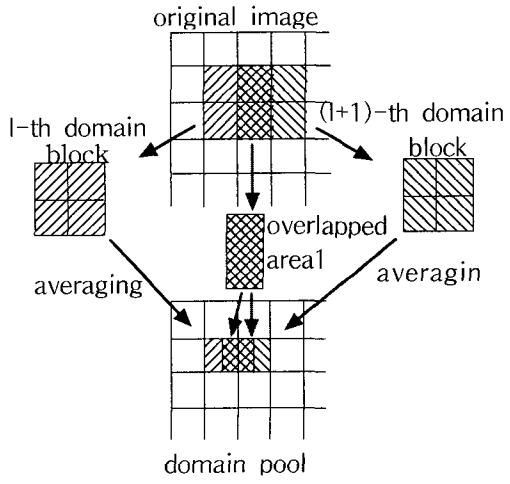

그림 4. 정의역블럭의 처리과정

Fig. 4. Processing procedure of domain blocks.

과정을 보인 것이다. 인접한 두 정의역블럭인 l번째와 $(l+1)$번째의 정의역블럭을 구성하는 데이터는 50%가 중첩되는 두 블럭을 축소변환한 것으로, 동일한 데이터를 포함하고 있다. 그림 3의 VLSI어레이에서16×16과 8×8 블럭크기에서 치역과 정의역블럭의 데이터를 순차적으로 번갈아 입력하는 대신 중복되는 정의역블럭의 데이터를 재사용하여 처리속도를 향상시킬 수 있다.

그림 5는 m번째 치역블럭과 정의역블럭들에 대한 데이터의존그래프이다. 데이터의존그래프의 인접한 열의 흰색과 회색노드의 한쌍에서 m번째 치역블럭과 l번째 정의역블럭의 쌍에 대한 연산이 수행된다. l번째 정의역블럭의 $D_{l,n/2}\sim D_{l,n-1}$은 $(l+1)$번째 정의역블럭의 $D_{l+1,0}\sim D_{l+1,n/2-1}$은 인접 두 정의역블럭의 중복된 영역으로 재사용 가능하다.

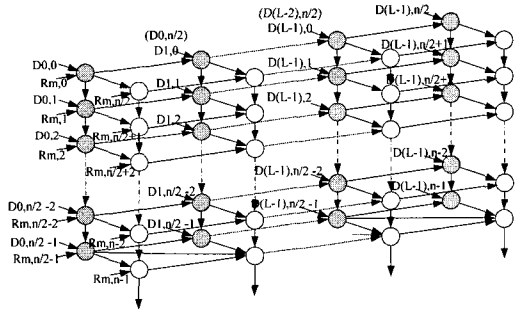

그림 5. 임의의 한 치역블럭과 정의역블럭간의 데이터 의존 관계(L : 정의역블럭의 수, n : 블럭 내의 픽셀수)

Fig. 5. Data dependence relations between an arbitrary range block and domain blocks (L : number of domain blocks, n : number of pixels for a block).

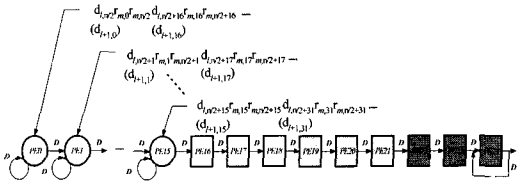

그림 6. 데이터 재사용에 의한 고속 일차원 어레이 및 데이터 입력 순서

Fig. 6. High-speed one dimensional array by data reusing and the ordering of input data.

정의역블럭의 입력 데이터를 재사용하기 위한 일차원 어레이 구조는 그림 6과 같다. 여기서 처리요소들의

수와 전체적인 어레이의 구성은 기존의 그림 3과 같지만, 처리요소의 내부구조와 입력데이터의 수와 순서가 다르다.

어레이에서 각 블럭크기에 대한 연산을 처리하기 위한 데이터(치역 및 정의역블럭을 구성하는 픽셀들)의 입력순서는 블럭의 크기를 입력한 후 정의역블럭의 픽셀값을 입력한다. 다음으로 이 정의역블럭의 픽셀값에 대응되는 치역블럭의 픽셀값을 입력하고, 일정한 치역블럭의 픽셀값이 차례로 입력된다. 이렇게 반복 수행하고, 최종결과가 설정한 MSE의 임계값을 만족하지 못하면 블럭의 크기를 변경시켜 앞의 과정을 반복하게 된다.

설계한 일차원 시스토릭 어레이는 정의역블럭의 데이터를 재사용함으로서, 8×8 블럭크기에서는 6단위시간, 16×16 블럭크기에서는 24단위시간 만에 한 쌍의 블럭에 대한 누적작업이 완료된다. 설계한 어레이는 데이터 입출력이 규칙적이고, 처리요소의 수와 구조가 영상의 크기에 독립적이며, 처리요소 내부의 연산장치도 서로 공유함으로써 VLSI 어레이의 구현에 적합한 구조를 갖는다.

2. 처리요소의 설계

설계한 일차원 시스토릭 어레이는 정의역 및 치역블럭의 픽셀값과 제곱의 값을 각각 누적하는 16개의 처리요소(*PE*0~*PE*15)와 *s*, *o*, MSE를 연산하고 최적의 값을 연산하는 9개의 처리요소(*PE*16~*PE*24)로 구성된다. 이 어레이는 기존의 어레이에서 정의역 및 치역블럭의 픽셀값들을 연산하는 16개의 처리요소의 내부구조에 레지스터 등 약간의 하드웨어가 추가되었으며, *s*, *o*, MSE를 연산하는 9개의 처리요소는 기존 어레이의 처리요소 내부구조와 동일하다.[13]

각 처리요소에는 정의역블럭과 치역블럭을 연산하는 장치와 입출력 값을 기억하는 레지스터들이 존재한다. 정의역블럭의 데이터를 재사용하기 위해서 제안한 어레이의 각 처리요소에는 레지스터 D가 추가된다. 정의역블럭의 데이터는 다음에 입력받을 치역블럭의 픽셀값과의 연산에서 재사용되므로 레지스터 D에 저장된다. 입력데이터들과 연산결과들은 다음 처리요소로 전달되도록 각 레지스터에 저장된다.

*PE*0의 내부구성은 그림 7과 같고, *PE*1부터 *PE*14까지의 내부구성은 그림 8과 같다. 성 등이 제안한 어레이 구조[13]에 비해 레지스터 D를 추가하여 입력 데이터의 재사용을 가능하게 한다.

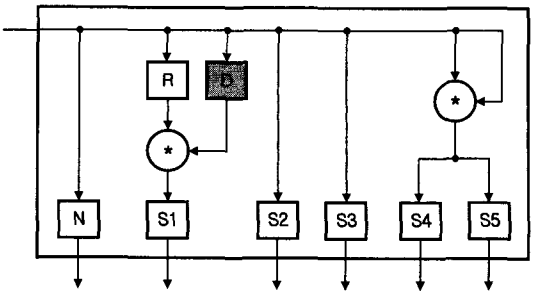

그림 7. PE0의 내부 조직
Fig. 7. Internal organization of PE0.

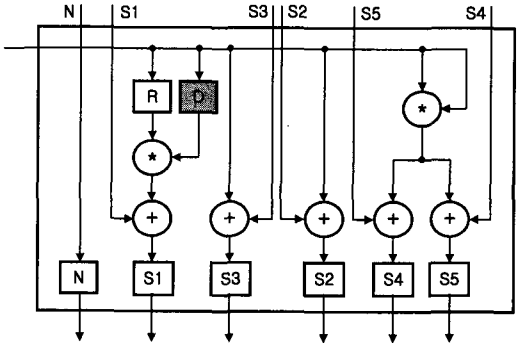

그림 8. PE1~PE14의 내부 조직
Fig. 8. Internal organization of PE1~PE14.

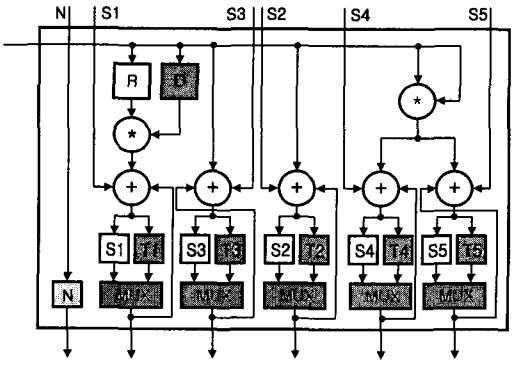

그림 9. PE15의 내부 조직
Fig. 9. Internal organization of PE15.

설계한 어레이는 블럭크기 4×4인 경우에 해당하며, 블럭크기가 8×8, 16×16인 경우, 반복연산을 통해서 그 결과를 얻을 수 있다. 그러므로 PE_a부분의 마지막 처리요소인 *PE*15는 그림 9와 같으며, 여기서는 기존에 저장되었던 연산결과에 현재의 연산결과를 누적하는 기능이 추가된다. 기본적인 동작은 앞의 처리요소들과 같으나, 8×8, 16×16 블럭크기에 대한 연산을 수행하

기 위해 반복해서 누적하는 부분이 존재한다. 연산결과들은 하나의 정의역블럭과 인접한 정의역블럭에 대한 값이 교대로 연산되므로 이 결과 값들은 각각의 해당 레지스터에 분류하여 저장한다. 그러므로 그림 9의 처리요소에서 연산된 결과들을 분류하여 누적하도록 멀티플렉서, 레지스터 $T1 \sim T5$가 추가된다.

s, o 및 MSE를 계산하는 처리요소들 즉 $PE17 \sim PE24$는 기존의 어레이 내부구조와 동일하다.[13]

Ⅳ. 평가 및 고찰

설계한 일차원 시스토릭 어레이 동작의 정확성을 검증하기 위해 512×512 크기의 Lenna 영상을 이용해서 컴퓨터 시뮬레이션을 수행하였다. 컴퓨터 시뮬레이션 프로그램은 Java Development Kit 1.3을 사용하여 작성하였다. 그림 10은 시뮬레이터의 스냅샷이다. 각 단계별로 처리요소 내의 레지스터의 값의 변화를 통해 설계한 어레이의 정확한 동작을 검증할 수 있었다.

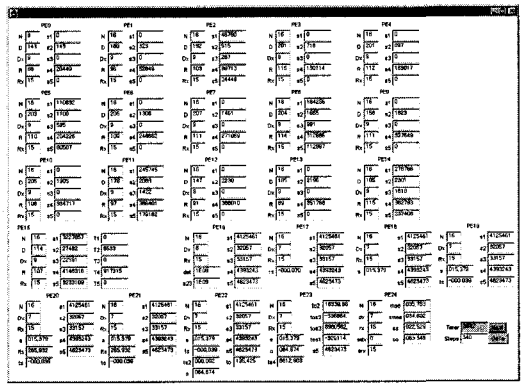

그림 10. 시뮬레이터의 스냅샷
Fig. 10. Snapshot of simulator.

본 논문에서 설계한 프랙탈 영상압축을 위한 시스토릭 어레이는 25개의 처리요소로 구성된다. 따라서 이 시스토릭 어레이를 이용한 영상에 대한 프랙탈 영상압축을 위한 계산시간은 식 (5)와 같다.

$$T = 24 + I_{n16} + I_{n8} + I_{n4} \tag{5}$$

이때 I_{n16}, I_{n8}, I_{n4}는 각각 블럭크기가 16×16, 8×8, 4×4일 때의 데이터 입력 횟수이다. 블럭크기가 4×4인 경우는 치역블럭의 값을 처리요소에 선적재시킨 후 정의역블럭의 데이터를 입력시키는 방법으로 기존의 어레이와 동일하다. 정의역블럭과 치역블럭의 데이터를 반복입력시키는 16×16, 8×8 블럭크기에 대해서는 데이터 입력회수를 줄여 시스토릭 어레이에서의 처리시간을 단축시킬 수 있다. 기존의 어레이에서 입력횟수, I_1는 식(6)과 같고[13], 본 논문에서 설계한 어레이에서의 입력횟수, I_2는 식(7)과 같다.

$$I_1 = \frac{B^2}{16} \cdot 2 \cdot \left(\frac{N-B}{B}\right)^2 \cdot \left(\frac{N}{B}\right)^2$$

$$I_2 = \frac{B^2}{16} \cdot \frac{3}{2} \cdot \left(\frac{N-B}{B}+1\right) \cdot \frac{N-B}{B} \cdot \left(\frac{N}{B}\right)^2$$

여기서 N은 영상의 크기를 나타내며, B는 블럭의 크기를 나타낸다. 식 (8)은 16×16과 8×8 블럭크기에 대한 입력횟수의 감소율로서 속도향상의 근사치로 평가할 수 있다.

$$S = 1 - \frac{I_2}{I_1} = 1 - \frac{3}{4} \cdot \frac{N}{N-B} = \frac{N-4B}{4(N-B)}$$

식(7)에서 예를 들어 N=512, B=16의 경우, S=0.226이 된다. 즉 최대 22.6%의 속도가 향상될 수 있다. 그림 11은 512×512 크기의 Lenna 영상에 대한 기존의 어레이와 본 논문에서 설계한 어레이를 이용하여 프랙탈 압축을 위한 계산시간을 임계값에 따라 비교한 것이다.[13] 임계값이 16인 경우, 19%의 성능향상을 얻을 수 있었다. 중복되는 데이터를 재사용하지 않는 4×4 블럭크기의 처리에 의해 식(8)의 최대 성능향상비를 얻지는 못했다. 그러나 하드웨어의 추가가 거의 없이 프랙탈 영상압축을 위한 전체 처리시간을 크게 줄일 수 있다.

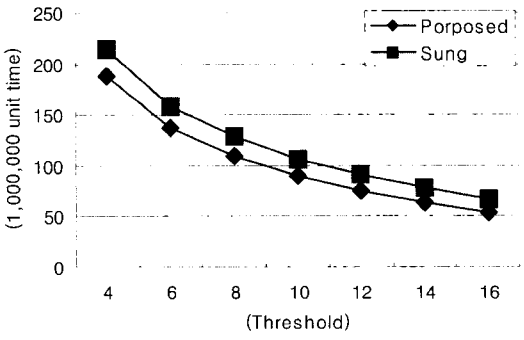

그림 11. 임계값에 따른 성능비교
Fig. 11. Performance comparison by thresholds.

V. 결 론

프랙탈 영상압축을 고속으로 수행하기 위해 VLSI 어레이 구조를 제안하였다. 기존의 제안된 일차원 VLSI 어레이에서 중첩되는 정의역블럭의 데이터들을 재사용하므로서 전체 연산에 필요한 데이터의 총 입력 횟수를 감소시키고, 이로 인한 전체 처리시간을 줄였다.

기존의 일차원어레이와 비교해서 처리요소의 수 및 연결구성에는 변경이 없고, 어레이로 입력되는 데이터의 순서를 재배치했으며, 어레이의 전반부의 처리요소들을 설계하였다. 즉 처리요소의 설계에서 정의역블럭의 데이터를 재사용하기 위하여 픽셀의 값을 연산하는 각 처리요소에 데이터를 저장하기 위한 하나의 레지스터가 추가되었다. 또한 픽셀의 값을 연산하는 마지막 처리요소에는 연산된 값들을 분류해서 누적하기 위해 멀티플렉서와 레지스터들이 추가되었다. 이 설계는 적은 하드웨어를 추가하여 기존의 어레이보다 영상과 블럭의 크기가 각각 512×512와 16×16이며 임계값이 16인 경우, 19%의 처리속도 향상을 가져왔다.

설계한 일차원 VLSI 어레이에 대한 동작의 정확성을 검증하기 위해 512×512 Lenna 영상을 사용하여 컴퓨터 시뮬레이션하였으며, 각 단위시간에 변하는 처리요소의 내부상태와 입출력되는 값들이 정확하게 변하는 것을 확인하였다. 본 논문의 연구결과는 프랙탈 영상압축의 실시간 처리를 위한 VLSI 시스템의 구현에 활용될 수 있을 것이다.

참 고 문 헌

[1] A. E, Jacquin, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations," IEEE Transactions on Image Processing, Vol. 1, No. 1, pp. 18-30, 1992.

[2] D. M. Monro and F. Dudbridge, "Fractal Block Coding of Images," Electron, Lett., Vol. 28, pp. 1053-1055, 1992.

[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] A. E. Jacquin, "Fractal Image Coding: A Review," Proceedings of the IEEE, Vol. 81, No. 10, pp. 1451-1465, Oct. 1993.

[5] Y. Fisher, Fractal Image Compression : Theory and Application, Springer-Verlag, Berlin. 1995.

[6] F. Ancarani, A. De Gloria, M. Olivieri, and C. Stazzone, "Design of an ASIC Architecture for High Speed Fractal Image Compression," Proceedings, 9th Annual IEEE International ASIC Conference and Exhibit, pp. 223-226, 1996.

[7] K. P. Acken, M. J. Irwin, and R. M. Owens, "A Parallel ASIC Architecture for Efficient Fractal Image Coding," Journal of VLSI Signal Processing, pp. 97-113, 1998.

[8] D. J. Jackson, S. J. Hannah, "Comparative analysis of image compression techniques," Proceedings of System Theory, SSST '93, 24th Southeastern Symposium on, pp. 513-517, Mar. 1993.

[9] S. Y. Kung, "VLSI Array Processors," Prentice Hall, 1988.

[10] C. C. Chen, "On the Selection of Image Compression Algorithms," Proceedings of Pattern Recognition, 14th International Conference on, Vol. 2, pp. 1500-1504, 1998.

[11] I. Andreopoulos, Y. A. Karayianris, T. Stouruaitis, "A hybrid image compression algorithm based on fractal coding and wavelet transform," Proceedings of Circuits and Systems, ISCAS 2000 Geneva, IEEE Int'l Symposium on, Vol. 3, pp. 37-40, 2000.

[12] 배성호, 박길흠 "웨이브렛 변환 영역에서의 유효계수 트리를 이용한 프랙탈 영상압축 방법," 전자공학회논문지, 제33권, B편 제11호, pp. 62-71, Nov. 1996

[13] 성길영, 우종호, "고속 프랙탈 영상압축을 위한 최적의 파이프라인 주기를 갖는 VLSI 어레이 구조 설계," 한국통신학회논문집, 제25권, 제5A호, pp. 702-708, May 2000

저 자 소 개

禹 鍾 鎬(正會員) 第34卷 C編 7號 參照

李 守 鎭(正會員)

1995년 2월 : 부산수산대학교 전자공학과 졸업(공학사). 1997년 2월 : 부경대학교 대학원 전자공학과 졸업(공학석사). 1999년 8월 : 부경대학교 대학원 전자공학과박사 수료.. <관심분야 : 병렬처리, 영상압축, 컴퓨터구조>

李 熙 珍(正會員)

1999년 2월 : 동의대학교 컴퓨터공학과 졸업(공학사). 2001년 9월 : 부경대학교 대학원 정보시스템협동과정 졸업(공학석사). <관심분야 : 병렬처리, 영상처리>

成 吉 永(正會員)

1980년 2월 : 경북대학교 전자공학과 졸업(공학사), 1985년 2월 : 건국대학교 대학원 전자공학과 졸업(공학석사). 2000년 8월 : 부경대학교 대학원 전자공학과 졸업(공학박사). 1995년~현재 : 경상대학교 정보통신공학과 교수, 해양산업연구소 연구원. <관심분야 : 영상압축, VLSI 어레이, 컴퓨터구조>