論 文

Vector-radix 2차원 고속 DCT의 VLSI 구현을 위한 효율적인 어레이 알고리듬

正會員 辛 卿 旭* 正會員 全 興 雨* 正會員 姜 龍 暹*

An Efficient Array Algorithm for VLSI Implementation of Vector-radix 2-D Fast Discrete Cosine Transform

Kyung Wook Shin*, Heung Woo Jeon*, Yong Seom Kang* *Regular Members*

要 約

본 논문에서는 vector-radix 2차원 고속 DCT(VR-FCT)를 VLSI 병렬계산하기 위한 효율적인 어레이 알고리듬을 제안하고, 이를 집적회로로 구현하기 위한 회로를 설계하였다. VR-FCT 알고리듬의 버터플라이 연산부분을 2차원 어레이에 매핑하여 이를 병렬 및 파이프라인 처리함으로써 VR-FCT 알고리듬의 고속성과 2차원 어레이의 병렬성 및 국부통신 특성을 동시에 이용할 수 있다는 특징을 갖는다. 제안된 구현 방식은 RCA 방식과는 달리 transposition 메모리가 필요치 않으며, 2차원 어레이의 구조적인 규칙성, 모듈성 및 국부연결성 등에 의해 회로설계 시간의 단축, 설계검증 및 설계변경 등이 용이하여 VLSI 구현에 매우 적합하다. 연산회로는 곱셈기를 사용하지않고 가산기만으로 설계하였으며, 2의 보수연산 대신에 Canonic-Signed Digit (CSD) 코드를 사용함으로써 약 30%의 가산횟수를 줄일 수 있었다. 제안된 방법의 DCT 연산과정을 C언어로 모델링하여 회로의 유한 레지스터 길이에 대한 연산정밀도를 분석하였다. 제안된 어레이 알고리듬의 시간성능은 (N×N) 2차원 DCT에 대해 $O(N+N_{NZD}\cdot\log_2 N)$의 시간 복잡도를 갖는다. 시뮬레이션 결과로 부터, $N_{NZD}=4$이고 50 MHz 클럭이 사용되는 경우, (8×8) DCT 계산에 약 0.88 μsec가 소요되며, 약 72×10^6pixels/sec의 연산성능이 예상된다.

ABSTRACT

This paper describes an efficient array algorithm for parallel computation of vector-radix two-dimensional (2-D) fast discrete cosine transform (VR-FCT), and its VLSI implementation. By mapping the 2-D VR-FCT onto a 2-D array of processing elements (PEs), the butterfly structure of the VR-FCT can be efficiently implemented with high concurrency and local communication geometry. The proposed array algorithm features architectural modularity, regularity and locality, so that it is very suitable for VLSI realization. Also, no transposition memory is required, which is inevitable in the conventional row-column decomposition approach. It has the time complexity of $O(N+N_{NZD}\cdot\log_2 N)$ for (N×N) 2-D DCT, where N_{NZD} is the number of non-zero digits in canonic-signed digit

*金烏工科大學 電子工學科
Dept. of Electronic Eng., Kumoh National University of Technology
論文番號 : 93-196

(CSD) code. By adopting the CSD arithmetic in circuit design, the number of additions is reduced by about 30%, as compared to the 2's complement arithmetic. The computational accuracy analysis for finite wordlength processing is presented. From simulation results, it is estimated that (8×8) 2-D DCT (with $N_{NZD}=4$) can be computed in about 0.88 μsec at 50 MHz clock frequency, resulting in the throughput rate of about 72 Mega pixels per second.

Ⅰ. 서 론

이산 코싸인 변환(discrete cosine transform : DCT)은 영상신호의 대역폭 압축을 위해 가장 일반적으로 사용되는 방법이며, JPEG, MPEG, CCITT H.261과 같은 국제 표준규격에서 영상압축 방식의 표준 알고리듬으로 채택되고 있고, 특히 디지탈 HDTV 방식에서도 DCT를 기본으로한 대역폭 압축방법이 연구되고 있다.[1-4]

2차원 DCT의 실시간 계산을 위해서는 전용 집적회로 소자의 사용이 필수적이며, 특히 실시간 영상처리 시스템의 개발을 위해서는 고성능 2차원 DCT 프로세서가 핵심부품으로 사용되어야 하므로 VLSI (very large scale integration) 기술을 이용한 2차원 DCT 프로세서의 개발에 대한 관심이 증대되고 있다.

DCT 프로세서의 구현방법은 크게 나누어 2차원 DCT의 분해특성을 이용하는 Row-Column Algorithm (RCA) 방식과 분해특성을 이용하지 않는 Non Row-Column Algorithm (NRCA) 방식으로 구분할 수 있다.[5]. RCA 방식은 2차원 영상에 대해 열방향(또는 열방향)의 1차원 DCT를 수행한 후, 그 결과를 matrix transposition하여 열방향(또는 행방향)의 1차원 DCT를 수행함으로써 2차원 DCT를 계산한다. 이 방식은 1차원 DCT 구조를 이용하여 2차원 DCT를 계산할 수 있다는 장점이 있으나, transposition 메모리가 부가적으로 필요하다는 단점이 있다. RCA 범주에 속하는 것으로는 prime factor 알고리듬[6], cyclic convolution 구조[7], 1차원 고속 알고리듬의 연산흐름도를 직접 매핑하는 방법[8], 그리고 분산연산(distributed arithmetic)을 이용하는 방법[9-11] 등이 발표되고 있다. 한편, vector-radix 분해[12,13], polynomial 변환[14] 등을 이용하는 고속 알고리듬, Winograd 변환을 이용한 알고리듬[15] 등은 NRCA 범주에 속하며, matrix transposition 메모리가 필요치 않다는 장점을 갖는다.

문헌에 발표된 대부분의 DCT 프로세서[7,8,10,16-18]들은 RCA 방식을 채택하고 있으며, 20-30 MHz 범위의 pixel rate를 가져 HDTV 신호처리등의 분야에서 요구되는 성능에는 미치지 못하는 것으로 평가되고 있다. 한편, 50MHz 이상의 pixel rate를 갖는 고성능 2차원 DCT 프로세서의 개발을 위해서는 고속 알고리듬의 개발과 함께 이를 실리콘에 효율적으로 매핑하기 위한 algorithm-specific 아키텍쳐의 고안, 그리고 아키텍쳐 레벨의 병렬성을 극대화하기 위한 최적화 방안 등이 종합적으로 연구되어야 한다.

일반적으로, 알고리듬의 효율적인 VLSI 구현을 위해서는 면적 복잡도와 시간 복잡도에 대한 면밀한 분석이 필요하며, DCT 알고리듬의 경우도 예외는 아니다. VLSI 구현을 위한 하드웨어 비용(즉, 칩면적)은 연산처리요소(즉, 승산기 및 가산기/감산기)의 갯수와 데이타 통신을 위한 배선면적에 의해 결정된다. 고속 알고리듬들은 연산횟수를 줄이는 댓가로 연산과 연산 사이의 데이타 통신구조가 복잡해지는 trade-off 관계를 갖는다. 따라서, 고속 DCT 알고리듬의 효율적인 VLSI 구현을 위해서는 규칙적인 연산구조를 갖는 알고리듬의 선택과 국부통신구조를 갖는 아키텍쳐의 설계, 그리고 알고리듬을 아키텍쳐에 효율적으로 매핑할 수 있는 매핑 알고리듬의 개발 등이 중요한 요소가 된다.

NRCA 범주에 속하는 vector-radix 2차원 고속 DCT (VR-FCT) 알고리듬은 RCA 범주의 알고리듬 보다 곱셈횟수가 25% 감소된 고속 알고리듬이며, Cooley-Turkey의 고속 퓨리어 변환(fast Fourier transform : FFT)의 버터플라이 연산흐름도와 동일한 규칙적인 연산구조를 갖는다. FFT의 버터플라이 구조는 규칙적인 연산구조를 갖는 장점은 있으나, 광역 데이타 통신을 수반하는 단점이 있어 VLSI 구현에 효율적이지 못한 것으로 인식되어 왔다. 그러나 최근의 연구결과에 의하면, FFT 버터플라이 연산구조를 처리요소들의 2차원 배열에 매핑하여 효율적인 VLSI 구현이 가능한 것으로 평가되었다.[19] 동일한 처리요소의 규칙적인 연결로 구성되는 2차원 어레이는 대규모

병렬처리 능력과 국부통신구조를 가지므로 VLSI 구현에 매우 적합한 구조이다.

본 논문에서는 VR-FCT 알고리듬의 연산특성과 데이타 이동 복잡도에 대한 분석을 토대로하여 VR-FCT 알고리듬의 고속성과 2차원 어레이의 구조적 특징을 동시에 이용하는 효율적인 VLSI 구현방법을 제안하고자 한다.

Ⅱ. Vector-Radix 2차원 고속 DCT(VR-FCT) 알고리듬

입력 데이타 {x(i, j)}에 대한 (N×N)-point 2차원 DCT의 계수 {X(u, v)}는 다음과 같이 정의된다.[1]

$$X(u,v)=\frac{4\varepsilon_u\varepsilon_v}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}x(i,j)\cdot\cos\frac{\pi(2i+1)u}{2N}\cdot\cos\frac{\pi(2j+1)v}{2N}\qquad(1)$$

단, $\varepsilon_u,\ \varepsilon_v=\begin{cases}1/\sqrt{2} & \text{for u, v}=0\\ 1 & \text{otherwise.}\end{cases}$

편의상, scale factor는 X(u, v)에 포함되며, $N=2^k$ (단, k는 양의 정수)이라고 가정한다. 아래의 식(2)의 index 매핑관계를 적용하여 식(1)을 다시 쓰면, 식(3)과 같이 된다.

$$f(i,j)=\begin{cases}x(2i,2j) & i=0,\cdots,N/2-1;\ j=0,\cdots,N/2-1\\ x(2N-2i-1,2j) & i=N/2,\cdots,N-1;\ j=0,\cdots,N/2-1\\ x(2i,2N-2j-1) & i=0,\cdots,N/2-1;\ j=N/2,\cdots,N-1\\ x(2N-2i-1,2N-2j-1) & i=N/2,\cdots,N-1;\ j=N/2,\cdots,N-1\end{cases}\qquad(2)$$

$$X(u,v)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}f(i,j)\cdot\cos\frac{\pi(4i+1)u}{2N}\cdot\cos\frac{\pi(4j+1)v}{2N}\qquad(3)$$

VR-FCT 알고리듬은 (N×N) 2차원 DCT를 4개의 (N/2×N/2) DCT로 분해하고, 각각의 (N/2×N/2)를 다시 4개의 (N/4×N/4)의 DCT로 분해하며, 이와 같은 분해과정을 (2×2) DCT가 얻어질 때까지 반복하여 순환구조를 얻는 것이다.[13]

식(3)에서 index (u, v)를 even-even, even-odd, odd-even, odd-odd 4개의 영역으로 분할하여 4개의 (N/2×N/2) DCT로 분할하고, 각각의 분할된 DCT에 대해 (2×2) DCT가 얻어질 때까지 동일한 분할과정을 반복하면 식(4)가 얻어진다.

i) even-even index part ;

$$X(2u,2v)=\sum_{i=0}^{N/2-1}\sum_{j=0}^{N/2-1}f^{(00)}(i,j)\cdot\cos\frac{\pi(4i+1)u}{2(N/2)}\cdot\cos\frac{\pi(4j+1)v}{2(N/2)}\qquad(4\text{ a})$$

ii) odd-even index part ;

$$X(2u+1,2v)=\sum_{i=0}^{N/2-1}\sum_{j=0}^{N/2-1}[2\cdot f^{(10)}(i,j)\cdot\cos\varphi_1]\cdot\cos\frac{\pi(4i+1)u}{2(N/2)}\cdot\cos\frac{\pi(4j+1)v}{2(N/2)}-X(2u-1,2v)\qquad(4\text{ b})$$

iii) even-odd index part ;

$$X(2u,2v+1)=\sum_{i=0}^{N/2-1}\sum_{j=0}^{N/2-1}[2\cdot f^{(01)}(i,j)\cdot\cos\varphi_2]\cdot\cos\frac{\pi(4i+1)u}{2(N/2)}\cdot\cos\frac{\pi(4j+1)v}{2(N/2)}-X(2u,2v-1)\qquad(4\text{ c})$$

iv) odd-odd index part ;

$$X(2u+1,2v+1)=\sum_{i=0}^{N/2-1}\sum_{j=0}^{N/2-1}[4\cdot f^{(11)}(i,j)\cdot\cos\varphi_1\cdot\cos\varphi_2]\cdot\cos\frac{\pi(4i+1)u}{2(N/2)}\cdot\cos\frac{\pi(4j+1)v}{2(N/2)}-X(2u-1,2v+1)-X(2u+1,2v-1)-X(2u-1,2v-1)\qquad(4\text{ d})$$

단, $f^{ab}(i,j)=f(i,j)+(-1)^a\cdot f(i+\frac{N}{2},j)+(-1)^b\cdot f(i,j+\frac{N}{2})+(-1)^{a+b}$

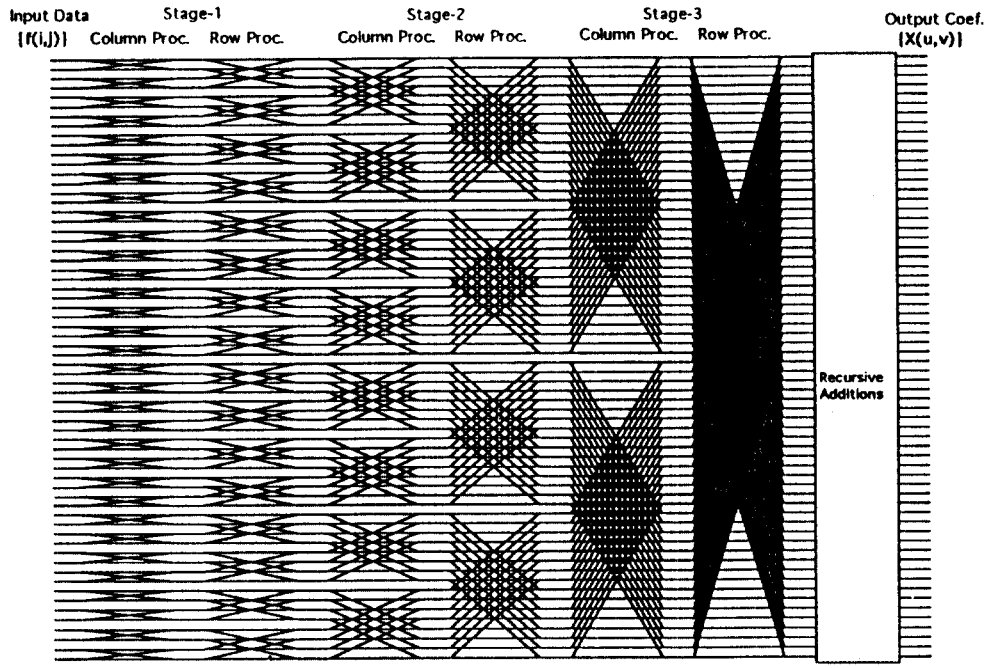


그림 1. (8×8) VR-FCT의 연산흐름도.
Fig. 1. The signal flow graph for (8×8) VR-FCT.

$$\cdot\ (i+\frac{N}{2},\ j+\frac{N}{2}) \text{ for } a,\ b = 0 \text{ or } 1$$

$$\cos\varphi_1 = \cos\frac{\pi(4i+1)}{2N},\quad \cos\varphi_2 = \cos\frac{\pi(4j+1)}{2N}$$

$$\text{for } i,\ j = 0,\ 1,\ \cdots,\ \frac{N}{2}-1.$$

식(4)로 부터 얻어지는 (8×8) VR-FCT의 연산 흐름도는 그림 1과 같다. 그림 1에 의하면, (8×8) VR-FCT는 $\log_2 8$ (즉, 3) stages의 버터플라이 연산과정과 후처리 가산과정에 의해 계산됨을 알 수 있다. 이를 일반화 시키면, (N×N) VR-FCT는 $\log_2 N$ stages의 버터플라이 연산과정과 후처리 가산과정에 의해 계산되며, 버터플라이 연산부분은 (N^2)-point 1차원 FFT와 동일한 연산구조를 갖는다. VR-FCT 알고리듬은 RCA 방식과 비교하여 연산횟수가 약 25% 감소된 고속 알고리듬이다.[13]

```
BEGIN  /* N = 2^k for positive integer k */
  FOR q=1 TO log2N DO IN PARALLEL for all i and j (0 ≤ i,j ≤ N-1)
     D_q = 2^(k-q)   /* Shuffling Distance */
     /* Column Processing */
     i_1 = i MOD 2^(k+1-q)
     IF (i_1 < 2^(k-q)) THEN
        move data f_(q-1)(i_1, j) to index point (i_1+D_q, j)
        PE(i_1, j) compute f_qe(i_1, j)=f_(q-1)(i_1, j)+f_(q-1)(i_1+D_q, j)
     ELSE
        move data f_(q-1)(i_1, j) to index point (i_1-D_q, j)
        PE(i_1, j) compute f_qe(i_1, j)=f_(q-1)(i_1-D_q, j)-f_(q-1)(i_1, j)
     ENDIF
     /* Row Processing */
     j_1 = j MOD 2^(k+1-q)
     IF (j_1 < 2^(k-q)) THEN
        move data f_qe(i, j_1) to index point (i, j_1+D_q)
        PE(i, j_1) compute f_q(i, j_1)=[f_qe(i, j_1)+f_qe(i, j_1+D_q)]·C_p
     ELSE
        move data f_qe(i, j_1) to index point (i, j_1-D_q)
        PE(i, j_1) compute f_q(i, j_1)=[f_qe(i, j_1-D_q)-f_qe(i, j_1)]·C_p
     ENDIF
  ENDFOR
END.
```

ALGORITHM-I. 2차원 어레이를 이용한 (N x N) 2-D VR-FCT 병렬계산 알고리듬
ALGORITHM-I. Proposed array algorithm for parallel computation of (N x N) 2-D VR-FCT using 2-D array.

Ⅲ. 2차원 어레이를 이용한 VR-FCT의 VLSI 병렬계산

Ⅱ장에서 VR-FCT 알고리듬이 규칙적인 버터플라이 연산과정과 후처리 가산과정으로 구성됨을 설명하였다. 본 장에서는 2차원 어레이를 이용하여 식(4)의 VR-FCT를 VLSI 병렬계산하기 위한 효율적인 방법을 제안하고자 한다.

식(4)에서 버터플라이 연산부분을 even-even, odd-even, even-odd, odd-odd 등 4개의 부분 index 영역으로 나누어 표현하면 식(5)와 같이 된다.

i) even-even index part :

$$f_q(i_1, j_1) = C_p \cdot [\{f_{q-1}(i_1, j_1) + f_{q-1}(i_1 + D_q, j_1)\} + \{f_{q-1}(i_1, j_1 + D_q) + f_{q-1}(i_1 + D_q, j_1 + D_q)\}] \quad (5\text{-a})$$

단, $C_p = 1$

ii) odd-even index part :

$$f_q(i_1 + D_q, j_1) = 2C_p \cdot [\{f_{q-1}(i_1, j_1) - f_{q-1}(i_1 + D_q, j_1)\} + \{f_{q-1}(i_1, j_1 + D_q) - f_{q-1}(i_1 + D_q, j_1 + D_q)\}] \quad (5\text{ b})$$

단, $C_p = \cos\{\pi(4i_1 + 1)(2^{q-2}/N)\}$

iii) even-odd index part :

$$f_q(i_1, j_1 + D_q) = 2C_p \cdot [\{f_{q-1}(i_1, j_1) + f_{q-1}(i_1 + D_q, j_1)\} - \{f_{q-1}(i_1, j_1 + D_q) + f_{q-1}(i_1 + D_q, j_1 + D_q)\}] \quad (5\text{-c})$$

단, $C_p = \cos\{\pi(4j_1 + 1)(2^{q-2}/N)\}$

iv) odd-odd index part :

$$f_q(i_1 + D_q, j_1 + D_q) = 4C_p \cdot [\{f_{q-1}(i_1, j_1) - f_{q-1}(i_1 + D_q, j_1)\} - \{f_{q-1}(i_1, j_1 + D_q) - f_{q-1}(i_1 + D_q, j_1 + D_q)\}] \quad (5\text{-d})$$

단, $C_p = \cos\{\pi(4i_1 + 1)(2^{q-2}/N)\} \cdot \cos\{\pi(4j_1 + 1)(2^{q-2}/N)\}$

한편, 식(5)에서 D_q는 q-번째 버터플라이 stage에서의 데이타 셔플링 거리를 나타내며, 다음과 같이 주어진다.

$$D_q = 2^{\log_2 N - q} \quad (6)$$

식(5)에서 index i_1, j_1은 버터플라이 stage q에 따라 달라지며, 각각 식(7)을 만족하는 index i, j의 조합으로 이루어진다.

$$\begin{aligned} i \quad MOD \quad & 2^{\log_2 N - q + 1} < 2^{\log_2 N - q} \\ j \quad MOD \quad & 2^{\log_2 N - q + 1} < 2^{\log_2 N - q} \end{aligned} \quad (7)$$

예를 들어, (8×8) DCT의 경우, sub-index set $\{(i_1, j_1)\}$는 다음과 같이 된다.

a) q = 1 일때 : i_1, j_1 = 0, 1, 2, 3
수, $\{(i_1, j_1)\}$ = {(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)}

b) q = 2 일때 : i_1, j_1 = 0, 1, 4, 5
수, $\{(i_1, j_1)\}$ = {(0,0), (0,1), (0,4), (0,5), (1,0), (1,1), (1,4), (1,5), (4,0), (4,1), (4,4), (4,5), (5,0), (5,1), (5,4), (5,5)}

c) q = 3 일때 : i_1, j_1 = 0, 2, 4, 6
수, $\{(i_1, j_1)\}$ = {(0,0), (0,2), (0,4), (0,6), (2,0), (2,2), (2,4), (2,6), (4,0), (4,2), (4,4), (4,6), (6,0), (6,2), (6,4), (6,6)}

식(5)를 고찰하여 보면, 중괄호 안의 연산 {·}은 row index가 D_q 만큼 떨어진 데이타 짝에 대해 이루어지며, 대괄호 안의 연산 [·]은 column index가 D_q 만큼 떨어진 데이타 짝에 대해 이루어짐을 알 수 있다. 따라서, 데이타 f(i, j)를 2차원 어레이에 매핑하면 중괄호 안의 연산 {·}은 열방향의 데이타 이동을 수반하고, 대괄호 안의 연산 [·]의 행방향의 데이타 이동을 수반한다. 결론적으로, 식(5)를 2차원 어레이에 매핑하여 처리하면 VR-FCT의 효율적인 병렬연산이 가능함을 알 수 있으며, 본 논문에서는 이와 같은 사실을 근거로하여 VR-FCT의 병렬처리 알고리듬을 제안하고자 한다.

(N×N) VR-FCT의 병렬계산을 위한 2차원 어레이는 N행×N열의 처리요소들로 구성되며, 데이타 f(i, j)의 index (i, j)는 처리요소의 index PE(i, j)와 직접적인 1:1 매핑관계를 갖는다. (8×8) VR-FCT

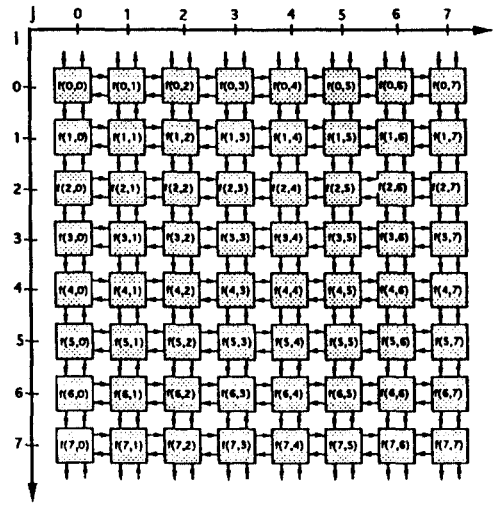


그림 2. (8×8) 2 D VR-FCT 구현을 위한 2차원 어레이 및 데이타 매핑.

Fig. 2. 2-D array and data mapping for implementing (8×8) 2-D VR-FCT.

를 위한 2차원 어레이는 그림 2와 같이 8행×8열의 2차원 어레이로 구성되며, 그림에서는 데이타 $\{f(i, j)\}$가 어레이에 매핑된 상태를 보이고 있다.

데이타가 어레이에 입력된 후, 데이타 셔플링과 버터플라이 연산이 $\log_2 N$ stages에 걸쳐 계산된 후, 후처리 가산과정을 거쳐 최종결과 $\{X(u, v)\}$가 출력된다. 각 버터플라이 연산 stage는 행방향 처리와 열방향 처리의 두단계 과정으로 이루어진다. 열방향 처리에서는 식(6)에 주어진 셔플링 거리 D_q 만큼 데이타가 열방향으로 이동된 후, 식(5)의 중괄호 안의 연산 $\{\cdot\}$ (즉, 버터플라이 데이타 짝에 대한 덧셈 또는 뺄셈연산)이 이루어진다. 한편, 행방향 처리에서는 셔플링 거리 D_q 만큼 데이타가 행방향으로 이동된 후, 식(5)의 대괄호 안의 연산 $[\cdot]$ (즉, 버터플라이 데이타 짝에 대한 덧셈 또는 뺄셈연산과 C_p값 곱셈연산)이 이루어진다.

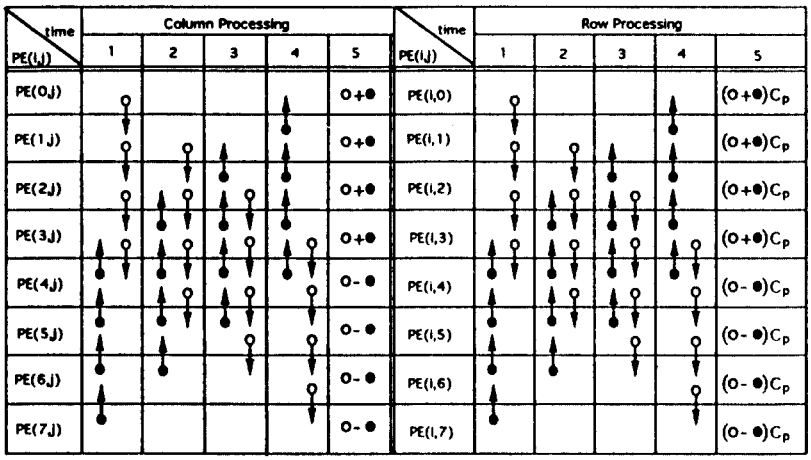
그림 2의 2차원 어레이상에서 식(5)가 계산되는 과정을 알고리듬으로 표현한 것이 ALGORITHM-I이며, 그림 3은 (8×8) VR-FCT가 ALGORITHM-I에 의해 계산되는 과정을 보인것이다. 버터플라이 연산 과정은 $\log_2 8 = 3$ stages로 구성되며, 각 연산 stage는 열방향 처리와 행방향 처리의 두단계 과정으로 이루어진다. stage-1의 연산과정은 다음과 같다. 열방향 처리 과정에서는 데이타가 열방향으로 셔플링거리 $D_1 = 2^{3-1} = 4$ 만큼 떨어진 처리요소로 이동된 후, 버터플라이 데이타 짝 (각 처리요소에 초기입력된 데이타와 이동된 데이타)에 대한 '+' 연산 또는 '−' 연산이 이루어진다. 행방향 처리과정에서는 열방향 처리에 의해 생성된 데이타가 행방향으로 셔플링거리 $D_1 = 2^{3-1} = 4$ 만큼 떨어진 처리요소로 이동된 후, 버터플라이 데이타 짝 (즉, 열방향 처리에 의해 생성된 각 처리요소내의 데이타와 이동된 데이타)에 대한 '+' 연산 또는 '−' 연산과 C_p값 곱셈이 이루어진다. stage-1의 데이타 셔플링은 셔플링 거리가 $D_1 = 4$ 이므로 4 클럭이 소요되며, 어레이를 구성하는 처리요소중 절반은 '+' 연산을 수행하고, 나머지 절반은 '−' 연산을 수행한다. stage-2와 3의 연산도 동일한 방식으로 이루어지며, stage-2의 셔플링거리는 $D_2 = 2$ 이고, stage-3의 셔플링거리는 $D_3 = 1$이 된다. 한편, 식(5)에서 C_p값은 버터플라이 stage q와 처리요소의 index (i, j)에 의해 결정된다.

Ⅳ. 회로설계

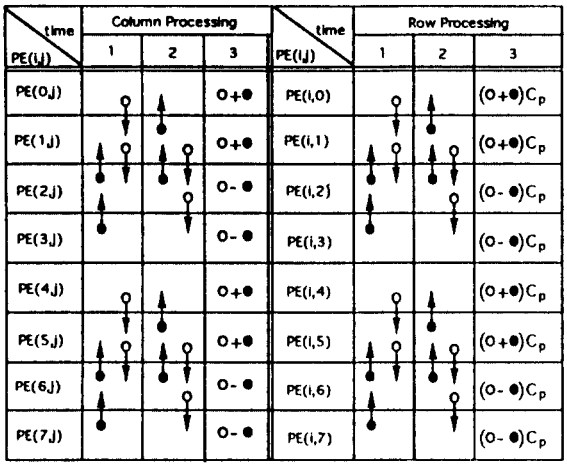
Ⅲ장에서 제안된 ALGORITHM-I은 그림 2와 같은 2차원 배열을 기본으로 하며, 배열을 구성하는 처리요소들은 동일한 기능을 수행하므로, 전체적인 회로설계는 기본 처리요소 하나의 설계로 단순화된다. 따라서, 본 장에서는 어레이를 구성하는 기본 처리요소의 회로설계에 대해 언급하고자 한다.

식(5)를 구현하기 위한 처리요소는 그림 4와 같이 4개의 기능블럭으로 구성된다. 즉, 인접한 처리요소와의 데이타 교환을 위한 데이타 셔플링 회로(P_Reg), 버터플라이 연산을 위한 연산회로(B_Ari), C_p값 저장을 위한 메모리(C_str), 그리고 처리요소의 동작을 제어하기 위한 제어신호 발생회로 등으로 구성된다.

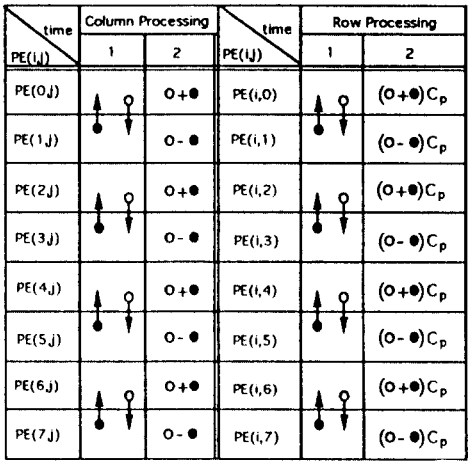
P_Reg 블럭은 인접한 처리요소와의 데이타 교환기능을 수행하며, 멀티플렉서와 파이프라인 레지스터로 구성된다. B_Ari 블럭은 식(5)를 계산하는 연산회로이며, 곱셈기를 사용하지 않고 가산기 만으로 설계하였다. 특히, 연산속도의 개선을 위해 canonic-signed digit (CSD)[20] 연산을 사용하였으며, 이와 같이 CSD 연산을 사용함으로써 2의 보수연산을 사용하는 경우 보다 약 30% 정도의 가산횟수를 줄일 수 있었다. C_p값은 처리요소의 위치와 버터플라이 stage q에 따라 고정되는 값이므로 CSD 코드로 변환되어 처리요소내에 저장된다.



(a) Stage 1



(b) Stage 2



(c) Stage 3

그림 3. (8×8) VR-FCT에대한 ALGORITHM I의 연산 과정
Fig. 3. Illustrations of the ALGORITHM I for (8×8) VR FCT.

처리요소를 구성하는 회로요소중 전체 칩의 성능(즉, 면적과 동작속도)에 가장 큰 영향을 미치는 부분은 B_Ari 블럭내의 가산기 회로이다. 가산기 회로는 ripple-carry 방식, carry lookahead 방식, carry select 방식, carry-save 방식등 여러가지 방식들이 있으나, 각 방식마다 면적과 시간성능 사이에 trade-off 관계가 존재하므로 적절한 선택이 필요하다. 본 논문에서는 가산기 회로의 선택기준으로 동작속도보다는 면적에 비중을 두고 ripple-carry 방식을 채택하였다. 그 이유는 전체 어레이를 단일 칩에 집적시키기 위해서는 처리요소의 면적을 최소화하는 것이 중요하기 때문이다.

설계된 회로의 논리기능을 논리 시뮬레이터 SILOS II를 이용하여 검증하였다. 그림 5는 $(A-B)\cdot C_p$의 연산이 처리요소에서 계산되는 과정을 시뮬레이션한 결과이다. Accumulator의 데이타 A = 52_H (단, 첨자 H는 16진수를 나타냄)와 Register_A의 데이타 B = CC_H에 대한 A-B 연산이 이루어지고, 연산결과인 86_H이 Accumulator에서 Register_A로 이동되며 Accumulator가 reset된다. 다음은 Register_A의 데이

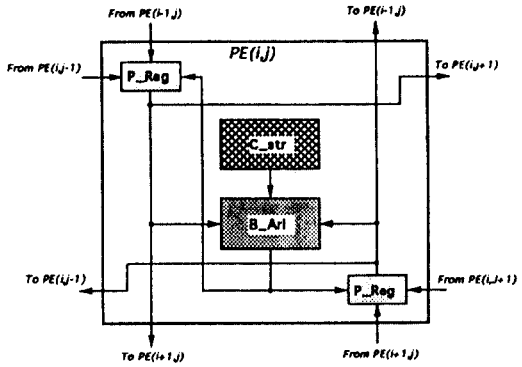


그림 4. 처리요소의 내부 구성도.
Fig. 4. Internal block diagram of processing element.

타 86_H과 CSD 코드로 표시된 C_p값 01010-100_2의 곱셈이 3클럭동안(즉, $N_{NZD}=3$) 연산되어 최종결과 $B7_H$이 Accumulator에 생성됨을 알 수 있다.

가산기는 그림 6의 회로를 사용하였다. 12-bit 가산기를 ISRC의 1.5μm N-well CMOS 공정 파라메터를 사용하여 SPICE 시뮬레이션한 결과는 그림 7과 같으며, SUM과 CARRY의 최대 지연 시간은 각각 13.8 nsec와 14.2 nsec로 나타났다. 동작클럭 주파수 결

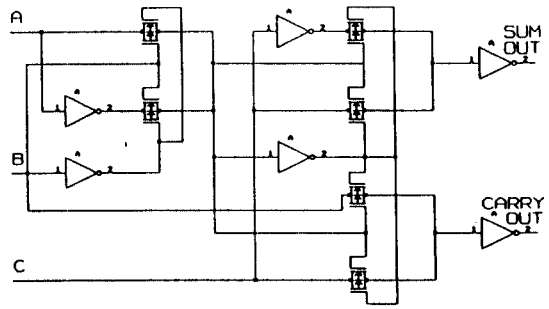


그림 6. 1-비트 전가산기 회로.
Fig. 6. 1-bit full adder circuit.

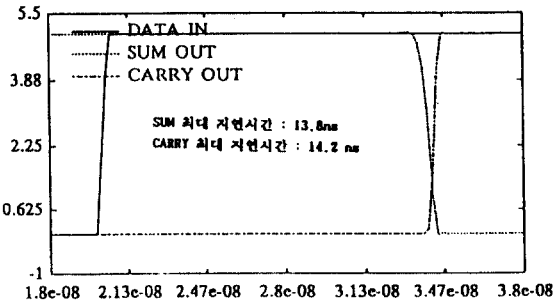


그림 7. 12-비트 ripple-carry 가산기의 SPICE 시뮬레이션 결과.
Fig. 7. SPICE simulation results for 12-bit ripple-carry adder.

```
      ***** SILOS OUTPUTS *****

      C SSSSSS LL LR C AA BBBBBBBB  DD  AA
      L EEEEEE DD DE I SS SSSSSSSS  RR  CC'
      O LLLLLL __ _S N 12 01234567  UU  CC
      C ______ AB AE               AA  __
      K AABBCC    CT                   HL
        121212    C                HL
TIME
   0  1 100000 10 01 0 00 10000000  80  **
  20  0 100000 00 01 0 00 10000000  80  **

 800  1 000000 00 11 1 01 10000000  CC  52
 820  0 000000 00 01 1 01 10000000  CC  52
 823  0 000000 00 01 1 01 10000000  CC  86   <-- 52H - CCH = 86H
 840  1 010000 10 01 0 00 10000000  CC  86
 882  1 000000 00 00 0 00 10000000  86  00   <-- Accumulator reset
 900  0 000000 00 00 0 00 10000000  86  00
 943  0 000000 00 01 0 00 01000000  86  C3   <-- 86H · 1/2 = C3H
 960  1 000000 00 11 0 00 00010000  86  C3
 983  0 000000 00 01 0 00 00010000  86  B3   <-- C3H + 86H · 1/8 = B3H
1000  1 000000 00 11 1 01 00000100  86  B3
1023  0 000000 00 01 1 01 00000100  86  B7   <-- B3H - 86H · 1/32 = B7H
1040  1 000000 00 11 0 00 10000000  86  B7
```

$(A-B)\cdot C_p$ 연산과정에 대한 시뮬레이션 결과임 (단, $A=52_H$, $B=CC_H$, $C_p=01010-100_2$)

그림 5. 처리요소 동작에 대한 논리 시뮬레이션 결과.
Fig. 5. Logic simulation results for processing element.

정을 위한 처리요소내의 최악 지연경로는 데이타 레지스터에서 부터 barrel shifter와 가산기를 거쳐 누산기에 이르는 경로이며, 이에 대한 SPICE 시뮬레이션 결과는 약 16 nsec로 나타났다. 이와 같은 시뮬레이션 결과를 토대로, 동작클럭 주파수를 50 MHz로 결정하였다.

V. 연산 정밀도 분석 및 성능평가

1. 연산정밀도 분석

DCT 프로세서 설계시에 중요하게 고려해야될 요소중의 하나는 연산 정밀도(computational accuracy)이다. DCT 연산시에 발생되는 오차는 그 평균값이 0이 되지 않으면 특정 위치에 집중되는 특성을 갖으며, 이는 역 DCT 과정에 누적되어 복원된 영상의 질에 치명적인 영향을 미치게 된다. 따라서, 유한 레지스터 길이(finite register length)를 갖는 DCT 프로세서의 경우에 연산 정밀도를 높이기 위해서는 오차의 평균값을 최소화하는 것이 중요하다.

DCT 과정에서 발생되는 연산오차는 프로세서의 구현방식에 따라 다르며, 일반적으로 다음과 같은 세 가지 원인에 기인한다. 첫째, 1차원 DCT의 반복계산에 의해 2차원 DCT를 계산하는 RCA 방식의 경우에는 1차원 DCT 연산회로의 출력단에서 발생되는 round-off 오차가 존재한다. 그러나, 본 논문의 경우에는 RCA 방식이 아닌 NRCA 방식을 사용하므로 이 오차는 발생되지 않는다. 둘째, 분산연산 방식을 사용하는 경우에는 메모리에 저장되는 partial product의 양자화 오차가 존재하며, 메모리에 저장되는 값의 word length에 따라 오차가 영향을 받는다. 그러나, 본 논문에서는 partial product 대신에 각 버터플라이 연산 stage에서 사용되는 계수값을 CSD 코드로 변환하여 메모리에 저장하는 방법을 사용하므로, 메모리에 저장되는 CSD 코드의 길이에 따라 오차가 영향을 받는다. 셋째, data path 내의 유한 레지스터 길이에 의한 오차가 발생한다. 일반적으로, data path 내의 레지스터 길이를 길게하면 연산오차는 감소하지만 칩면적이 증가하므로, 연산 정밀도와 칩면적 사이에는 trade-off 관계가 존재한다. 따라서, 최적의 레지스터 길이를 결정하기 위해서는 연산과정에 대한 정밀도 분석(accuracy analysis)이 필요하다.

본 논문에서는 연산정밀도 분석을 위하여 Ⅲ장에서 제안된 ALGORITHM-I에 의한 DCT 및 역 DCT 계산과정을 C 프로그램으로 모델링하여 CSD 코드의 길이(N_{CSD})와 data path 내의 레지스터 길이(N_{REG})에 따른 연산오차를 시뮬레이션하였다. 여러가지 시험영상에 대하여 원래 영상을 DCT한 후, 이를 다시 역 DCT하여 재생영상을 만들고 원래 영상과의 오차를 계산하여 연산정밀도를 분석하였다. 그림 8은 시험영상 'Girl'에 대해 N_{CSD}와 N_{REG}를 변화시키면서 peak signal to noise ratio(PSNR)을 계산한 결과이다. 그림 8로 부터, data path내의 레지스터 길이가 16 비트 이상인 경우와 CSD 코드의 길이가 12비트 이상인 경우에는 연산 정밀도가 더이상 개선되지 않음을 알 수 있다. 그림 9(a)의 시험영상 'Girl'에 대해 N_{CSD} = 11 비트로 고정시킨 상태에서 N_{REG} = 13, 15, 16 비트 각각에 대해 시뮬레이션하여 복원된 영상이 그림 9(b), (c), (d)이다.

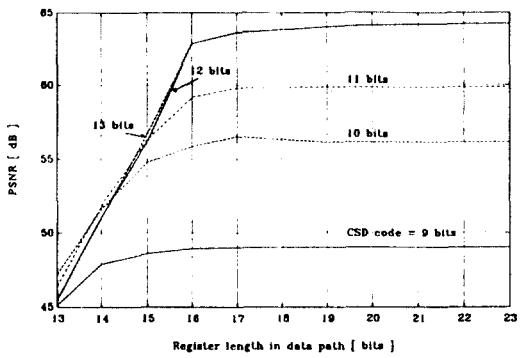


그림 8. 시험영상 'Girl'에 대한 연산정밀도 분석결과.
Fig. 8. Simulation results of computational accuracy for test image 'Girl'.

2. 성능평가

본 논문에서 제안된 DCT 구현방법의 시간성능은 아래의 식(8)이 나타내는 총 연산시간 T_{total}로 표현된다.

$$T_{total} = T_{shuffling} + T_{butterfly} \quad (8)$$

식(8)에서 $T_{shuffling}$와 $T_{butterfly}$는 각각 $\log_2 N$ stages에 걸친 데이타 셔플링 시간과 버터플라이 연산시간을 나타낸다. 데이타 셔플링은 행방향 처리와 열방향 처리에서 모두 이루어지므로, 총 셔플링 시간은 식(6)이 나타내는 셔플링 처리를 $\log_2 N$ stages 만큼 합한 것의 두배가 되며, 식 (9-a)와 같이 주어진다. 한편, 버터플라이 연산에서 '+' 또는 '-' 연산은 행방

(a) original image

(b) N_{REG} = 13 bits

(c) N_{REG} = 15 bits

(d) N_{REG} = 16 bits

그림 9. N_{REG} = 13, 15, 16 비트로 DCT 및 역DCT 처리하여 복원된 영상. (N_{CSD} = 11 비트)

Fig. 9. Reconstructed images of DCT/IDCT processing with N_{REG} = 13, 15, 16 bits (N_{CSD} = 11 bits)

향 처리와 열방향 처리에서 모두 이루어지는 반면에 C_p값 곱셈은 행방향 처리에서만 이루어지므로, 총 버터플라이 연산시간은 식 (9-b)와 같이 주어진다.

$$T_{shuffling} = 2T \cdot \sum_{q=1}^{\log_2 N} (D_q + 1) = 2T \cdot \sum_{q=1}^{\log_2 N} (2^{\log_2 N - q} + 1)$$

$$= 2T \cdot (N + \log_2 N - 1) \quad (9\text{-a})$$

$$T_{butterfly} = (4 + N_{NZD}) \cdot \log_2 N \cdot T \quad (9\text{-b})$$

식(9)에서 N_{NZD}는 CSD 코드내의 non-zero digit의 갯수를 나타내며, T는 클럭주기이다. 식(9)를 식(8)에 대입하여 정리하면,

$$T_{total} = [\, 2(N-1) + (6 + N_{NZD}) \cdot \log_2 N \,] \cdot T \quad (10)$$

따라서, 본 논문에서 제안된 DCT 구현방법은 (N×N) 2차원 DCT에 대해 $O(N + N_{NZD} \cdot \log_2 N)$의 시간 복잡도를 갖는다. 예를 들어 (8×8) DCT의 경우, $T_{total} = (32 + 3 \cdot N_{NZD}) \cdot T$ 클럭이 소요되며, $N_{NZD} = 4$이고 50 MHz 클럭이 사용되는 경우, 0.88 μsec가 소요되며, 약 72×10^6 pixels/sec의 연산성능이 예상된다.

Ⅵ. 결 론

본 논문에서는 vector-radix 2차원 고속 DCT(VR FCT)를 VLSI 병렬계산하기 위한 효율적인 어레이 알고리듬을 제안하고, 이를 집적회로로 구현하기 위한 회로를 설계하였다. VR-FCT 알고리듬의 버터플라이 연산부분을 2차원 어레이에 매핑하여 이를 병렬 및 파이프라인 처리함으로써 VR-FCT 알고리듬의 고속성과 2차원 어레이의 병렬성 및 국부통신 특성을 동시에 이용할 수 있다는 특징을 갖는다. 제안된 구현방식은 RCA 방식과는 달리 transposition 메모리가 필요치 않다. 또한, 2차원 어레이의 구조적 특성인 규칙성, 모듈성 및 국부연결성 등에 의해 VLSI 구현에 매우 적합하다는 장점을 갖는다. 전체 어레이는 동일한 처리요소들의 2차원 배열로 구성되므로, 회로 설계는 처리요소 하나의 설계로 단순화되며, 따라서 설계시간이 단축되고 설계검증 및 설계변경 등이 용이하다. 제안된 구현 방법의 연산과정을 C 프로그램으로 모델링하여 회로의 유한 레지스터 길이에 대한 연산정밀도를 분석하였다. 설계된 회로의 논리동작을 논리 시뮬레이터인 SILOS로 검증하였다. 연산회로는 메모리를 기본으로 하는 분산 연산 방식 대신에 계수값을 CSD 코드로 변환하여 곱셈기없이 가산기만으로 설계하였으며, CSD 코드를 사용함으로써 2의 보수연산에 비해 약 30%의 가산횟수를 줄일 수 있었다. 제안된 DCT 구현방법의 시간성능은 (N×N) 2차원 DCT에 대해 $O(N + N_{NZD} \cdot \log_2 N)$의 시간 복잡도를 갖는다. $N_{NZD} = 4$이고 50 MHz 클럭이 사용되는 경우에, (8×8) DCT 계산에 약 0.88 μsec가 소요되며, 약 72×10^6 pixels/sec의 연산성능이 예상된다. 따라서, HDTV 등 실시간 영상처리를 위한 2차원 DCT 프로세서의 기본 아키텍처로 이용가능할 것으로 평가된다.

참 고 문 헌

1. K.R. Rao and P. Yip, Discrete Cosine Transform ; Algorithms, Advantages, Applications, Academic Press, Inc., 1990.
2. G.K. Wallace, "The JPEG still-picture compression standard," Communications of the ACM, vol.34, no.4, pp.31-44, Apr., 1991.
3. D.Le Gall, "MPEG : A video compression standard for multimedia applications," Communications of the ACM, vol.34, no.4, pp.47-58, Apr., 1991.
4. K.B. Benson and D.G. Flink, HDTV Advanced television for the 1990s, McGraw-Hill, 1991.
5. S. Wolter, D. Birreck and R. Laur, "Classification for 2D DCT's and a new architecture with distributed arithmetic," 1991 IEEE International Symposium on Circuits and Systems (ISCAS'91), pp.2204-2207, 1991.
6. N.I. Cho and S.U. Lee, "DCT algorithms for VLSI parallel implementations," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol.ASSP 38, no.1, pp.121-127, Jan., 1990.
7. J. Carlach, P. Penard and J. Sicre, "TCAD : a 27MHz 8×8 discrete cosine transform chip," 1989 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '89), pp.2429-2432, 1989.

8. A. Artieri et al, "A one chip VLSI for real time two-dimensional discrete cosine transform," 1988 IEEE International Symposium on Circuits and Systems (ISCAS'88), pp.701-704, 1988.
9. B. Sikstroem, et al. "A high speed 2-D discrete cosine transform chip," Integration VLSI Journal 5, pp.159-169, 1987.
10. M.T. Sun, T.C. Chen and A.M. Gottlieb, "VLSI implementation of a 16×16 discrete cosine transform," IEEE Trans. on Circuits and Systems, vol.CAS-36, no.4, pp.610-617, Apr., 1989.
11. U. Sjoestroem et al, "Discrete cosine transform chip for real-time video applications," 1990 IEEE International Symposium on Circuits and Systems (ISCAS'88), pp.1620-1623, 1990.
12. M.A. Haque, "A two-dimensional fast cosine transform," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol.33, no.6, pp.1532-1536, Dec. 1985.
13. S.C. Chan and K.L. Ho, "A new two-dimensional fast cosine transform algorithm," IEEE Trans. on Signal Processing, vol.39, no.2, pp. 481-485, Feb., 1991.
14. P. Duhamel, C. Guillemot, "Polynomial transform computation of the 2-D DCT," 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'90), pp.1515-1518, 1990.
15. M. Yan and J.V. McCanny, "VLSI architecture for computing the 2-D DCT," International Conference on Systolic Arrays, pp.411-429, 1989.
16. L. Matterne et al, "A flexible high performance 2-D discrete cosine transform IC," 1989 IEEE International Symposium on Circuits and Systems (ISCAS'89), pp.618-621, 1989.
17. A. M. Chiang, "A video rate CCD two-dimensional cosine transform processor," Visual Commun. and Image Process. II, SPIE, vol.845, pp. 2-5, 1987.
18. T. Denayer et al, "A 50 MIPS multiprocessor chip for image processing," 1988 IEEE Custom Integrated Circuit Conference (CICC'88), pp.8.3.1-8.3.4, 1988.
19. M. K. Lee, K. W. Shin and J.K Lee, "A VLSI array processor for 16-point FFT," IEEE J. of Solid-State Circuits, vol.SC-26, no.9, pp.1286-1292, Sep., 1991.
20. K. Hwang, Computer arithmetic : Principles, Architecture and Design, John Wiley & Sons, 1979.
21. P/C SILOS : Logic Simulator, SIMUCAD, Inc., 1988.

※본 논문은 1992년도 한국학술진흥재단 지원 지방대신진과제 학술연구조성비의 지원에 의한 연구결과임

辛 卿 旭(Kyung Wook Shin) 정회원
1961년 10월 26일생
1980년 3월 ~ 1984년 2월 : 한국항공대학 전자공학과(공학사)
1984년 3월 ~ 1986년 2월 : 연세대학교 대학원 전자공학과(공학석사)
1986년 3월 ~ 1990년 8월 : 연세대학교 대학원 전자공학과(공학박사)
1990년 9월 ~ 1991년 6월 : 한국전자통신연구소 반도체연구단 선임연구원
1991년 7월 ~ 현재 : 금오공과대학교 전자공학과 조교수
※주관심분야 : VLSI Signal Processing, VLSI Architectures, ASIC Design

姜 龍 遷(Yong Seom Kang) 정회원
1965년 1월 11일생
1986년 3월 ~ 1992년 2월 : 금오공과대학교 전자공학과(공학사)
1992년 3월 ~ 현재 : 금오공과대학교 대학원 전자공학과 재학중
※주관심분야 : VLSI 설계, ASIC 설계

全 興 雨(Heung Woo Jeon) 정회원
1956년 10월 30일생
1975년 3월 ~ 1980년 2월 : 한국항공대학 전자공학과(공학사)
1980년 3월 ~ 1982년 2월 : 고려대학교 대학원 전자공학과(공학석사)
1982년 3월 ~ 1988년 8월 : 고려대학교 대학원 전자공학과(공학박사)
1989년 3월 ~ 현재 : 금오공과대학교 전자공학과 조교수
※주관심분야 : VLSI 설계, CAD