한국통신학회논문지 '12-11 Vol.37A No.11
http://dx.doi.org/10.7840/kics.2012.37A.11.993

리프팅 기반 2차원 이산 웨이블렛 변환 필터의 효율적인 VLSI 구조

박 태 구◆, 박 태 근°,

Efficient VLSI Architecture for Lifting-Based 2D Discrete Wavelet Transform Filter

Taegu Park◆, Taegeun Park°

요 약

본 논문에서는 리프팅 기반의 하드웨어 효율이 100%가 되는 2차원 이산 웨이블릿 변환 필터 구조를 제안한다. 전체구조는 (9,7) 필터를 적용하였으며, 필터의 길이에 따라 확장 및 축소가 가능하다. 본 연구에서 제안하는 새로운 스케줄링은 블록기반으로 수행하며 하위 레벨을 수행할 조건이 충족되면 바로 해당레벨을 수행하므로 중간 값을 저장해야 하는 시간이 짧아지며, 따라서 이를 위한 레지스터 양을 최소화할 수 있다. 제안된 스케줄링에 맞는 입력을 조절하기 위해 그에 적절한 DFC(Data Format Converter)와 DCU(Delay Control Unit)구조를 설계하였다. 입력 영상이 $N\times N$이고 m을 필터 길이라고 할 때, 필요한 저장공간은 $2mN$이다. 인접한 4개의 데이터를 동시에 입력 받아 동시에 행 방향과 열 방향 DWT를 수행하므로 J가 분해 레벨이라고 할 때 총 $N^2(1-2^{-2J})/3$ 사이클이 소요된다.

Key Words : lifting-based DWT, VLSI architecture, 2D DWT, data scheduling

ABSTRACT

In this research, we proposed an efficient VLSI architecture of the lifting-based 2D DWT (Discrete Wavelet Transform) filter with 100% hardware utilization. The (9,7) filter structure has been applied and extendable to the filter length. We proposed a new block-based scheduling that computes the DWT for the lower levels on an "as-early-as-possible" basis, which means that the calculation for the lower level will start as soon as the data is ready. Since the proposed 2D DWT computes the outputs of all levels by one row-based scan, the intermediate results for other resolution levels should be kept in storage such as the Data Format Converter (DFC) and the Delay Control Unit (DCU) until they are used. When the size of input image is $N\times N$ and m is the filter length, the required storage for the proposed architecture is about $2mN$. Since the proposed architecture processes the 2D DWT in horizontal and vertical directions at the same time with 4 input data, the total period for 2D DWT is $N^2(1-2^{-2J})/3$.

Ⅰ. 서 론

정보통신의 발달이 괄목할만한 성장을 이룸에 따라 소비자들은 더욱 고품질의 서비스를 요구하고

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2011-0009513).
※ 본 연구는 2011년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음(M-2011-B0002-00195).
◆ 주저자 : 가톨릭대학교 정보통신전자공학과 VLSI시스템설계 연구실, skytogo@hanmail.net, 학생회원
° 교신저자 : 가톨릭대학교 정보통신전자공학과 VLSI시스템설계 연구실, parktg@catholic.ac.kr, 정회원
논문번호 : KICS2012-08-364, 접수일자 : 2012년 8월 20일, 최종논문접수일자 : 2012년 11월 13일

있으며, 그에 따른 데이터의 양은 증가하게 되었다. 디지털 영상정보를 있는 그대로 저장매체에 저장할 경우 많은 양의 메모리를 필요로 할 뿐만이 아니라 전송 시 많은 시간과 비용을 필요로 한다. 따라서 압축하는 부호화 기술과 복원하는 복호화 기술은 정보화 시대의 필수 불가결한 요소이다.

영상압축에는 DCT(Discrete Cosine Transform)과 DWT(Discrete Wavelet Transform)의 두 가지가 널리 이용된다. 블록기반으로 행해지는 DCT는 요구되는 메모리의 용량이 적고 블록기반 움직임 보상과 움직임 예측의 이용에 있어서 조화가 좋지만 8×8 블록 단위로 처리하기 때문에 블록 경계에서의 블록효과 문제가 발생한다. DWT는 이미지 또는 프레임 전체에 대해서 변환하므로 DCT와는 다르게 블록효과가 없다. 또한 JPEG보다 훨씬 적은 비트레이트에서 같은 압축률을 제공한다. 따라서 저속의 비트레이트에서 고품질의 영상을 제공해준다[1]. 웨이블릿 압축 작업은 먼저 이미지를 분석하여, 그것을 수신 측에서 복원할 수 있는 수학적 표현으로 변환함으로써 이루어지며 영상 및 음성 신호처리분야에 많이 응용되고 있으며[2,3], 이미지 압축과 음성분석, 패턴 인식, 그리고 컴퓨터 비전 등 다양한 분야에서 응용되어 JPEG2000의 표준으로 채택되었다[4,5].

리프팅 방식은 DWT 처리방식의 일종이며 웨이블릿 변환필터를 수학적인 표현을 이용해 변환함으로써 단순한 연속된 행렬식의 연산으로 나타낼 수 있다[6]. 복잡한 기본적인 웨이블릿 변환을 이용한 필터링 기법에 비해서 메모리양의 요구가 적고 순방향과 역방향의 변환이 동일한 구조를 이루는 장점을 가지고 있다. 최근 리프팅 기반의 DWT를 위한 다양하고 효율적인 VLSI구조가 많이 연구되었다[7-15].

(4,2) 필터를 사용한 2D DWT구조가 Ferretti와 Rizzo에 의해 제안되었다[7]. 행 방향을 따라 update와 predict를 수행하기 위한 두 개의 필터와 열 방향을 따라 update와 predict를 수행하기 위한 두 개의 필터로 구성되었으며, 파이프라인을 지원하기 위해 4개의 버퍼를 사용한다. Recursive Pyramid Algorithm(RPA)과 중첩구조(folded architecture)를 혼합함으로써 하드웨어 효율을 더욱 높을 수 있는 Folded RPA 구조[8]가 Jung 등에 의해 제안되었다. Lian 등에 의해서 제안된 행 프로세서는 1D folded 구조이며 열방향을 따라 기존의 방식대로 연산이 수행된다[9]. 성능과 하드웨어 복잡도의 Tradeoff를 감안한 리프팅 기반의 2D DWT 구조가 제안되었다[10]. Generalized 2D DWT 구조[11]가 Andra 등에 의해 제안되었으며 필터의 길이에 대해 확장성을 가진다. 이것은 하나 또는 두 개의 스텝을 가지는 리프팅 구조에 대해 두 가지 범주의 아키텍쳐를 지원한다. 하지만 많은 양의 메모리가 필요하다는 단점이 있다. 하드웨어 효율을 높이기 위해 인터리브 방식을 사용한 2D Recursive 구조[12]가 Liao등에 의해 제안되었다. 행 방향 프로세서와 열 방향 프로세서는 1D Recursive 구조와 유사하며 메모리가 적게 들지만 제어가 복잡하다. 라인 기반의 구조에 병렬적 기법과 파이프라인 기법을 적용하여 고속처리가 가능한 구조가 Xiong 등에 의하여 제안되었다[13]. 콘볼루션 기반의 recursive 2D DWT 구조가 Cheng 등에 의하여 제안되었는데 성능은 개선되었지만 비교적 큰 용량의 on-chip 메모리가 필요하다[15].

본 논문에서는 리프팅 기반의 하드웨어 효율이 100%가 되며 필터 길이 m에 대해 확장이 가능한 2차원 DWT필터 구조를 제안한다. 전체구조는 (9,7) 필터를 기본으로 하였으며, 행과 열을 동시에 처리하기 위해 1차원 DWT 필터 4개를 이용한다. 필요한 메모리를 최소화하기 위해 전체 이미지를 $2^J \times 2^J$의 블록으로 나눈 후, 하나의 블록 내에서 다시 $2^{J-1} \times 2^{J-1}$의 블록으로 나누며 이러한 과정을 2×2의 서브블록까지 진행한다. 스캔 과정은 최하위 블록을 기준으로 지그재그 스캔 한 후 상위 블록으로 넘어가 지그재그 스캔 하는 과정을 최상위 블록까지 수행하는 방식이다. 이러한 스케줄링을 적용할 시 중간 값을 저장하는 시간이 짧아지므로 필요한 레지스터의 양을 최소화 할 수 있다는 장점을 가지고 있다. 스케줄링의 적용을 위해서 입력데이터를 조절해주는 DFC(Data Format Converter)와 중간 데이터의 지연 시간을 조절해주는 DCU(Delay Control Unit)를 설계한다.

본 논문의 구성은 다음과 같다. 먼저 Ⅱ장에서는 리프팅 알고리즘의 이론적 배경과 구조에 대해서 설명하고, Ⅲ장에서는 제안한 스케줄링과 구조에 대해서 설명한다. 그리고 Ⅳ장에서는 성능을 비교하였으며 마지막으로 Ⅴ장은 본 논문의 결론이다.

Ⅱ. 리프팅 기반 웨이블릿 알고리즘

리프팅 방식은 DWT의 한 방법으로 수학적인 기

법을 이용하여 계산과정을 간단히 하는 것이 기본 원리이다[6]. DWT의 기본적인 연산을 표현해보면 아래의 식과 같이 나타낼 수 있다.

$$\tilde{p}(z)=\begin{bmatrix}\widetilde{h_e}(z)\,\widetilde{h_o}(z)\\ \widetilde{g_e}(z)\,\widetilde{g_o}(z)\end{bmatrix},$$

$$p(z)=\begin{bmatrix}h_e(z)\,g_e(z)\\ h_o(z)\,g_o(z)\end{bmatrix}$$

$\tilde{h}$와 $\tilde{g}$는 각각 저주파, 고주파 필터이며, $\tilde{p}(z)$는 순방향, $p(z)$는 역방향 DWT를 나타낸다.

DWT의 특성인 biorthogonality를 이용하여 위의 식을 분해하여 표현하면 아래와 같이 나타낼 수 있다.

$$\tilde{p}(z)=\left\{\prod_{i=1}^{n}\begin{bmatrix}1 & \widetilde{s_i}(z)\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \widetilde{t_i}(z) & 1\end{bmatrix}\right\}\begin{bmatrix}K & 0\\ 0 & \dfrac{1}{K}\end{bmatrix}$$

$$p(z)=\left\{\prod_{i=1}^{n}\begin{bmatrix}1 & 0\\ -\widetilde{t_i}(z) & 1\end{bmatrix}\begin{bmatrix}1 & -\widetilde{s_i}(z)\\ 0 & 1\end{bmatrix}\right\}\begin{bmatrix}\dfrac{1}{K} & 0\\ 0 & K\end{bmatrix}$$

K는 스케일 상수이며, n은 (5,3) 필터의 경우 1이며 (9,7)필터의 경우 2이다. 첫 번째의 update 단계와 두 번째의 predict 단계를 처리한 후 스케일링을 거친다. Update 단계를 통해 짝수 값을 계산하며 predict단계를 통해 홀수 값을 계산한 후 결과의 상호 연관성을 이용하여 계산과정을 줄일 수 있게 된다. 각 스텝에서의 연산은 다음의 과정을 따른다.

predict step

$$y_{2n+1}=x_{2n+1}+a(x_{2n}+x_{2n+2})$$

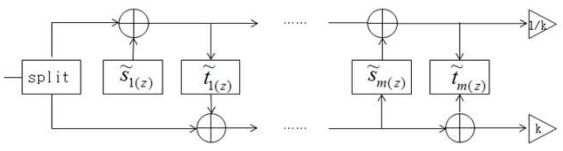
update step

$$y_{2n}=x_{2n}+a(y_{2n-1}+y_{2n+1})$$

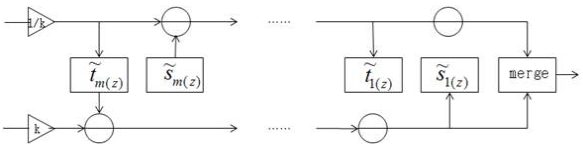
scaling step $\quad Z_{2n}=Ky_{2n},$

$$Z_{2n+1}=K^{-1}y_{2n+1}$$

이를 블록 다이어그램으로 나타내면 그림 1과 같다.



(a) 순방향 변환 (a) Forward Transform



(b) 역방향 변환 (b) Inverse Transform

그림 1. 리프팅 기반 순방향/역방향 DWT
Fig. 1. Lifting based forward/inverse DWT

(9,7) 필터의 경우 행렬식은 다음과 같다.

$$\tilde{p}(z)=\begin{bmatrix}1 & a(1+z^{-1})\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ b(1+z) & 1\end{bmatrix}\begin{bmatrix}1 & c(1+z^{-1})\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ d(1+z) & 1\end{bmatrix}\begin{bmatrix}K & 0\\ 0 & 1/K\end{bmatrix}$$

위의 식에서 a=-1.586134342, b=0.005298011854, c=0.8829110762, d=-0.4435068522, K=1.149604398 이다[8]. (9,7) 필터의 데이터 흐름을 그림 2와 같이 나타낼 수 있다.

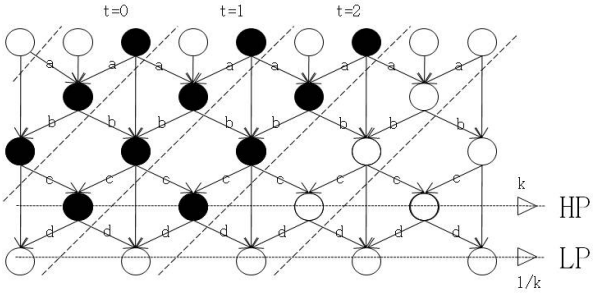


그림 2. (9,7) 필터의 데이터 흐름도
Fig. 2. Data flow of (9,7) filter

Ⅲ. 제안된 2D DWT 구조

3.1. 전체 구조 및 스케줄링

본 논문에서는 하드웨어 효율이 100%가 되는 2차원 DWT를 구현하기 위해 이를 위한 DFC와 DCU 구조를 제안한다. 4개의 1D DWT를 이용하여 행 방향과 열 방향의 DWT를 동시에 수행한다. 그림 3은 본 연구에서 제안하는 리프팅 기반 2차원 DWT 구조를 나타낸다. 하나의 1D-HDWT^{U}에는 위쪽 행의 인접한 두 개의 데이터(U_{UU}와 U_{UL})를 입력해준다. 다른 하나의 1D-HDWT^{L}에는 아래쪽 행의 인접한 두 개의 데이터(U_{LU}와 U_{LL})를 입력한다. 1D-HDWT^{U}의 결과 중 저주파출력 L과 1D-HDWT^{L}의 결과 중 저주파출력 L이

1D-VDWT^{U}에 입력되어 2D DWT의 결과 중 저주파(LL)과 고주파(LH)를 생성한다. LL은 하위레벨을 계산할 때 입력으로 이용되기 위해 DFC에 저장된다. DFC는 데이터 값을 저장하기 위한 레지스터와 데이터 분배를 위한 멀티플렉서로 구성되었으며, 외부입력과 저장된 데이터(LL) 중 적절한 값을 출력해준다. 1D-DWT필터에는 각각 DCU 모듈이 포함되어 있으며, 각 필터 단에 위치해 있다. 외부입력과 각 필터 단에서의 출력 데이터를 저장함으로써 스케줄링에 따른 수행시간에 알맞은 값의 출력이 가능하도록 하는 역할을 한다.

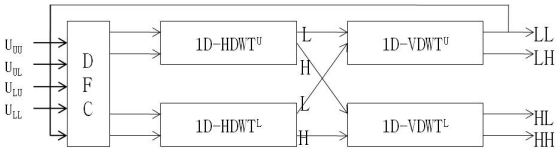
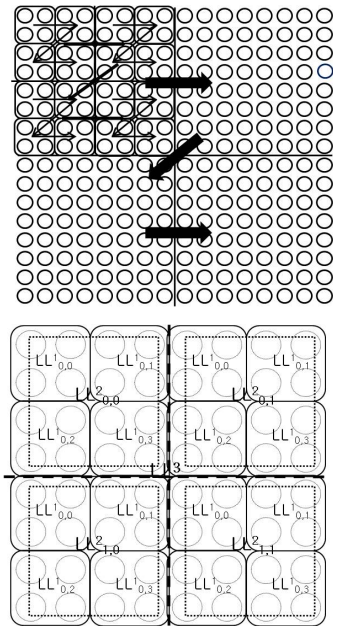


그림 3. 제안한 리프팅 기반 2D DWT 구조
Fig. 3. Proposed lifting based 2D DWT architecture



(a) 데이터 스케줄링 (b) DWT 출력결과
(a) Data scheduling (b) DWT output]

그림 4. 16x16 입력 영상 스캔 순서 및 DWT 출력결과
Fig. 4. 16x16 Input image scan order and DWT output

본 논문에서는 하드웨어 효율이 100%가 되기 위한 새로운 블록기반의 스케줄링을 제안한다. J 레벨의 해상도를 갖는 2차원 DWT필터는 $N\times N$ 이미지를 갖는 2차원 데이터를 $2^{J}\times 2^{J}$블록 단위로 나눈다. 여기에서 N은 2의 배수의 정수 값을 가진다고 가정한다. 모두 $X\times X$개의 블록으로 나누어지며, 이때 $X=N/2^{J}$이다. 나누어진 블록에서 다시 $2^{J-1}\times 2^{J-1}$의 서브블록으로 나누며 이러한 과정을 2×2의 서브블록까지 진행한다. 데이터의 스캔 과정은 최하위 블록을 기준으로 지그재그 스캔 한 후 상위 블록으로 이동해 지그재그 스캔을 거친다. 이러한 방법의 스캔은 레지스터 값을 유지하는 시간을 최단으로 하여 인접한 4개의 데이터를 이용하여 바로 상위 레벨을 수행함으로써 최소의 메모리만을 요구하는 장점을 지닌다. 이를 그림 4(a)에 나타내었으며, 그림 4(b)는 스캔에 따른 출력결과를 보여준다.

3.2. 제안된 스케줄링에 따른 레지스터 할당

그림 5는 3레벨 DWT 연산 시에 사용되는 레지스터를 나타내는 그림이다. J는 수행하는 레벨을, t_r과 t_c는 각각 행 방향과 열 방향의 연산 시간을 의미한다. 각 칸에는 같은 시간에 사용되는 인접한 데이터 4개가 입력된다. 각 데이터는 그림 5에 나타낸 것과 같이 UU, UL, LU, LL로 정의한다. $R^{j}_{a,b}$에서 첨자 j는 레벨을 가리키며 a와 b는 출력된 데이터의 행과 열을 나타내는 것으로서 0,0은 UU의 데이터를 저장하고, 0,1은 UL의 데이터를 저장하며, 1,0과 1,1은 각각 LU와 LL의 데이터를 저장한다.

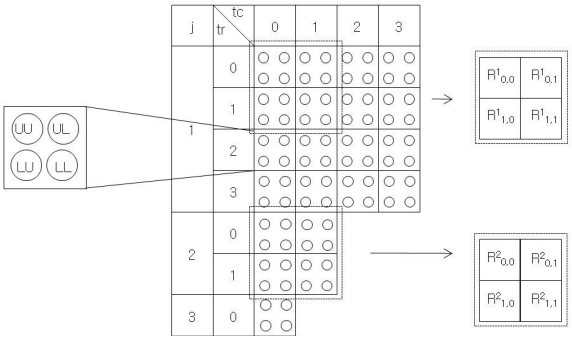


그림 5. 레지스터 할당 방법
Fig. 5. Register allocation scheme

3레벨을 수행한다고 가정할 때의 과정을 설명하면 다음과 같다. 필터는 행과 열을 동시에 처리하기 위해 동시에 4개의 데이터를 입력 받는다. 가장 먼저 (j,t_r,t_c)=(1,0,0)의 시간이 수행된다. 출력결과는 (2,0,0)시간에 필요한 데이터 중 UU을 의미하며, 하위레벨을 수행하기 위한 데이터로서 임시 저장된다. 1레벨의 결과이며 UU 데이터이기 때문에

$R^1_{0,0}$에 저장된다. 다음으로 (1,0,1)의 시간에 수행되는 출력결과는 (2,0,0)시간의 데이터 중 UL을 의미한다. 1레벨의 결과이며 UL데이터이기 때문에 $R^1_{0,1}$에 저장된다. 이와 같은 방식으로 그림 4와 같은 스캔과정을 따라 3레벨까지 수행한다.

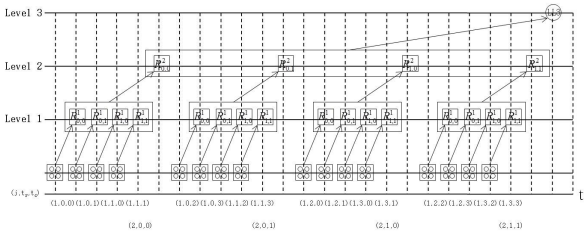


그림 6. 수행시간에 따른 레지스터 할당
Fig. 6. Timing diagram for register allocation

그림 6은 스케줄링에 따라 이용되는 레지스터 그룹을 나타낸 것이며, 그림에서 보는 것과 같이 인접한 4개의 데이터가 생성되면 바로 상위레벨을 수행하기 때문에 각 레벨당 하나의 레지스터 그룹이 반복되어 사용되는 것을 볼 수 있다. 따라서 각 레벨당 필요한 레지스터의 수는 4개이며 총 J 레벨까지 수행된다고 할 때 $4(J-1)$개의 레지스터가 필요하다.

3.3. 리프팅 1D DWT변환 필터

본 논문에서 제안하는 리프팅 기반의 1D DWT 구조는 그림 7에 나타내었다. 구조는 (9,7) 필터를 기준으로 하였으며 필터 길이에 따라 확장과 축소가 가능하다.

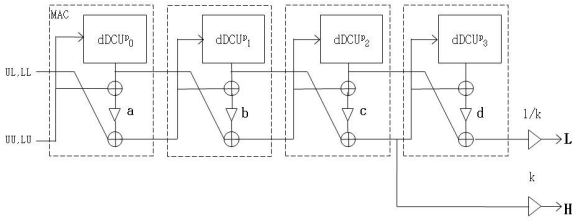


그림 7. 제안된 리프팅 기반 1D DWT 필터
Fig. 7. Proposed lifting based 1D DWT filter

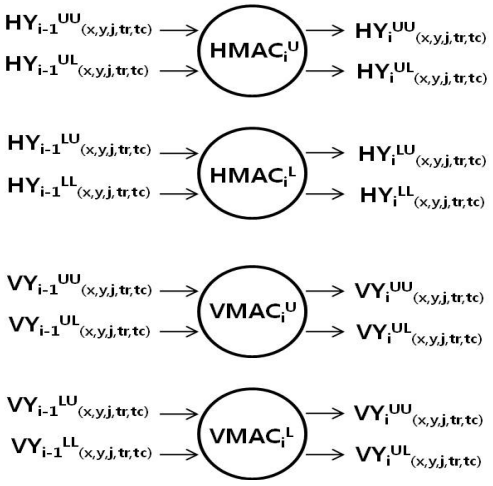


그림 8. 각 필터 단의 입출력 정의
Fig. 8. Input/output definition for each filter module

(9,7) 필터의 경우 총 4개의 필터 단을 가지고 있으며, (5,3) 필터의 경우에는 2개의 필터 길이를 갖는다. 하나의 필터 단은 두 개의 곱셈기와 하나의 덧셈기를 포함하며 연산에 필요한 중간 데이터를 저장하기 위하여 DCU 모듈을 가지고 있다. 그림의 $dDCU_i^p$에서 d는 행 방향필터일 경우 H이며 열 방향 필터일 경우 V이다. 첨자 p는 두 개의 HDWT필터 혹은 두 개의 VDWT필터 중 위쪽의 필터이면 U, 아래쪽의 필터이면 L이다. 첨자 i는 몇 번째 필터 단인지를 나타낸다. 첫 번째 필터 단을 제외한 나머지 MAC에서의 입출력 관계는 그림 8과 같이 정의한다. 괄호 안의 x와 y는 전체 이미지를 $2^J \times 2^J$의 블록으로 나누었을 때의 블록위치를 나타내는 행과 열의 값이다. j는 수행되는 레벨을 가리키며, t_r과 t_c는 수행되는 레벨에서의 행과 열에 대한 시간을 의미한다.

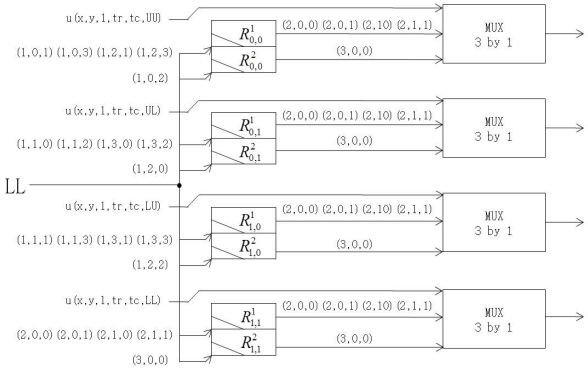


그림 9. DFC 구조($J=3$)
Fig. 9. DFC structure($J=3$)

3.4. DFC(Data Format Converter) 구조

HDWT에는 외부입력 또는 각 레벨에서 출력된 LL이 입력된다. 외부입력은 저장될 필요가 없지만 각 레벨의 수행결과인 LL은 하위레벨 수행 시 사용하기 위하여 저장되어야 한다. 따라서 데이터의 저장과 필터로의 입력을 조절해주기 위해 DFC 구조를 설계한다. 그림 9는 3레벨에서의 DFC 구조를 나타낸다. 그림에서 각각의 멀티플렉서들은 다음에 연결된 두 개의 필터에 각각 두 개씩의 입력 데이터를 전달해 준다. 괄호 안의 숫자는 (j,t_r,t_c)를 나타내며 해당 타이밍에 출력되고 저장된다. 다음의 두 식은 DFC를 설계하는데 이용된다.

$$HMAC_0^U(x,y,j,t_r,t_c)=$$
$$\begin{cases} u(x,y,j,t_r,t_c,UU); j=1,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \\ u(x,y,j,t_r,t_c,UL); j=1,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \\ LL(x,y,j,t_r,t_c,UU); j\geq 2,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \\ LL(x,y,j,t_r,t_c,UL); j\geq 2,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \end{cases}$$

$$HMAC_0^L(x,y,j,t_r,t_c)=$$
$$\begin{cases} u(x,y,j,t_r,t_c,LU); j=1,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \\ u(x,y,j,t_r,t_c,LL); j=1,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \\ LL(x,y,j,t_r,t_c,LU); j\geq 2,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \\ LL(x,y,j,t_r,t_c,LL); j\geq 2,t_r=0,...,2^{J-1}-1,t_c=0,...,2^{J-1}-1 \end{cases}$$

3.5. DCU(Delay Control Unit) 구조

MAC에서 DCU는 (x,y,j,t_r,t_c)에 따라 알맞은 데이터를 입력해주는 역할을 한다. 최상위 블록을 기준으로 했을 때, 행에 대해서는 연속적인 블록으로 수행되지만 열에 대해서는 연속적인 블록으로 진행되는 것이 아니기 때문에 행에서의 데이터 유지시간보다 오랜 시간 동안 값을 유지하고 있어야 한다. 따라서 행에 대한 DCU보다 많은 양의 메모리를 필요로 한다. 다음 식은 DCU를 설계하기 위한 식으로 이용된다.

$$HY_i^{UL}(x,y,j,t_r,t_c)=HDCU_{i-1}^U(x,y,j,t_r);j\geq 1,t_r=0,...,2^{J-j}-1$$
$$HY_i^{LL}(x,y,j,t_r,t_c)=HDCU_{i-1}^L(x,y,j,t_r);j\geq 1,t_r=0,...,2^{J-j}-1$$
$$VY_i^{UL}(x,y,j,t_r,t_c)=VDCU_{i-1}^U(x,y,j,t_c);j\geq 1,t_c=0,...,2^{J-j}-1$$
$$VY_i^{LL}(x,y,j,t_r,t_c)=VDCU_{i-1}^L(x,y,j,t_c);j\geq 1,t_c=0,...,2^{J-j}-1$$
$$HDCU_i^U(x,y,j,t_r,t_c)=HY_{i-1}^{UU}(x,y,j,t_r,t_c);j\geq 1,t_r=0,...,2^{J-j}-1$$
$$HDCU_i^L(x,y,j,t_r,t_c)=HY_{i-1}^{LU}(x,y,j,t_r,t_c);j\geq 1,t_r=0,...,2^{J-j}-1$$
$$VDCU_i^U(x,y,j,t_r,t_c)=VY_{i-1}^{UU}(x,y,j,t_r,t_c);j\geq 1,t_c=0,...,2^{J-j}-1$$
$$VDCU_i^L(x,y,j,t_r,t_c)=VY_{i-1}^{LU}(x,y,j,t_r,t_c);j\geq 1,t_c=0,...,2^{J-j}-1$$

본 연구에서 사용된 (9,7) 필터의 경우는 총 4단의 길이로 구성되어있기 때문에 데이터의 유지를 위해 1D DWT 필터의 경우 DCU도 4단이 필요하다. 그림 10는 3레벨의 경우에 $HDCU^U$ 구조를 나타내며 $HDCU^L$도 같은 구조로 구성할 수 있다.

그림11은 3레벨에서의 $VDCU^U$ 구조를 나타내는 것으로서 한 열의 구조는 $HDCU$와 같지만, 데이터의 유지시간이 X 블록만큼 길기 때문에 $HDCU$보다 X의 배수만큼 많은 것을 알 수 있다. $VDCU^L$도 같은 구조로 구성된다.

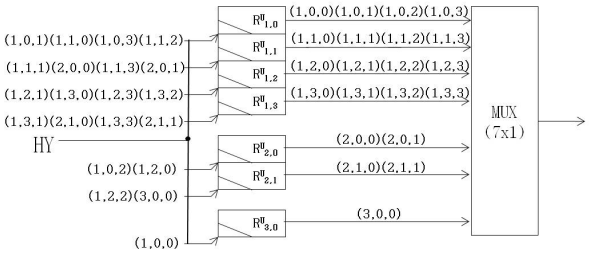


그림 10. $HDCU^U$ 구조($J=3$)
Fig. 10. $HDCU^U$ structure($J=3$)

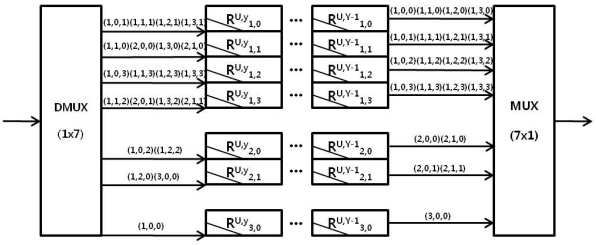


그림 11. $VDCU^U$ 구조($J=3$)
Fig. 11. $VDCU^U$ structure($J=3$)

Ⅳ. 설계 및 성능 분석

제안된 아키텍쳐는 (9,7) 리프팅 필터를 기반으로 하여 verilogHDL로 설계하여 검증되었으며, 적절한 비트 값을 할당해 주기 위해 각 계수간의 정수와 소수부분의 비트 수를 달리하였다. 즉 계수 a와 스케일링 상수 K만 정수 1비트, 소수 10비트로 할당하였으며, 정수부분이 필요없는 나머지는 소수 11비트로 할당하였고 출력은 모두 16비트로 할당하였다. 표 1은 제안한 구조와 기존의 리프팅 기반 2D DWT 구조를 성능과 하드웨어 복잡도 측면에서 비교 분석하였으며 이 때, 모든 구조는 (9,7) 필터를 기준으로 하였다.

본 논문에서 제안하는 스케줄링은 기존의 블록기반에서 서브블록 개념을 도입함으로써 스케줄링의 복잡도는 증가하였지만 메모리 사용을 최소로 할 수 있게 되었다. m개의 필터단을 갖는 1D DWT

는 하나의 MAC당 두개의 곱셈기와 하나의 덧셈기 및 하나의 DCU를 가지며 다른 구조에 비교하여 하드웨어 효율이 100%를 나타낸다. 입력 영상이 $N \times N$이고 m을 필터 길이라고 할 때, 필요한 저장공간은 $2mN$이다. 인접한 4개의 데이터를 동시에 입력 받아 동시에 행 방향과 열 방향 DWT를 수행하므로 J가 분해 레벨이라고 할 때 총 $N^2(1-2^{-2J})/3$ 사이클이 소요된다. 블록단위로 처리되므로 3레벨을 가정할 때 영상 데이터가 4개씩 입력되어 최종데이터가 출력되는데 걸리는 시간인 레이턴시(latency)는 그림5에서와 같이 21클록이다.

표1에서 보는 바와 같이 처리 성능(cycles)면에서 기존의 구조 중에서 가장 우수한 처리 성능을 보인다. 연산기(곱셈기, 덧셈기, 등) 부분에서는 행과 열을 동시에 처리하기 때문에 기존의 구조보다 비슷하거나 많은 하드웨어를 사용하지만 메모리 부분에서는 행열 변환 메모리(transpose memory)가 필요 없으므로 가장 적은 메모리로 2D DWT 연산이 가능하다.

표 1. 하드웨어 복잡도 및 성능비교(N^2: 영상크기)
Table 1. Hardware complexity and performance comparison(N^2: 영상크기)

Architectures	multipliers /shiters	adders	Memory		cycles	hardware utilization
			on-chip	off-chip		
Andra[11]	6	8	N^2	0	N^2+N	100%
Liao[12]	12	16	$10N$	0	N^2+N	50% ~ 66.7 %
Barua[13]	12	16	$7N$	$N^2/4$	$2N^2/3$	100%
Xiong(HA)[14]	18	32	$5.5N$	$N^2/4$	$N^2/3$	100%
Cheng[15]	24	76	$30N$	$2N$	$N^2/3$	100%
proposed	16	32	$8N$	0	$N^2/3$	100%

V. 결 론

본 논문에서는 데이터의 효율적인 압축을 위해 리프팅 기반의 2차원 이산 웨이블릿 필터 구조에 대해서 연구하였다. 100%의 하드웨어 효율을 얻기 위해 인접한 4개의 데이터를 동시에 입력 받았으며 4개의 1D DWT를 사용하여 행과 열을 동시에 처리할 수 있다. (9,7) 필터 기반의 DWT변환 필터를 설계하였으며, 필터의 길이에 따라서 확장 및 축소가 가능하다. 본 논문에서 제안된 블록 기반 스케줄링은 하위레벨의 값 계산에 필요한 중간 값을 유지하는 시간을 짧게 해주어 레지스터의 가용성을 높여주며 따라서 메모리를 최소화할 수 있는 장점을 지닌다. 이와 같은 스케줄링에 대해 적절한 데이터를 입력해주기 위해 DFC와 DCU를 설계하여 데이터의 입출력을 조절하였으며, 제안한 구조는 verilogHDL로 설계하고 검증하였다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여준 IDEC에 감사드린다.

References

[1] S. Mallat, “A theory for multiresolution signal decomposition: The wavelet representation,” *IEEE Trans. Pattern Anal. And Machine Intell.*, vol. 11, no. 7, pp. 674-693, Jul. 1989.

[2] R. Kronland-Martinet, J. Morlet, and A. Grossmann, “Analysis of sound patterns through wavelet transforms,” *Int. J. Pattern Recognition and Artificial Intelligence*, vol. 1, no. 2, pp. 273-302, Aug. 1987.

[3] S. Mallat, “Multifrequency channel decompositions of images wavelet models,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12, pp. 2019-2110, Dec. 1989.

[4] T. Ryan, L. Sanders, H. Fisher, and A. Iverson, “Image compression by texture modeling in the wavelet domain,” *IEEE Trans. Image Process.*, vol. 5, no. 1 pp. 26-36, Jan. 1996.

[5] A. Skodras, C. Christopoulos, and T. Ebrahimi, “JPEG2000: The Upcoming Still Image Compression Standard,” *Proceedings of Conference of Pattern Recognition*, vol. 22, no. 12, pp. 359-366, Oct., 2000.

[6] T. Acharta and C. Chakrabarti, “A Survey on Lifting-based Discrete Wavelet Transform Architectures,” *J. of VLSI Signal Process.*, vol. 42, no. 3, pp. 321-339, Mar. 2006.

[7] M. Ferreti and D. Rizzo, “A parallel architecture for the 2D discrete wavelet

transform with Integer Lifting Scheme," *J. of VLSI Signal Process.*, vol. 28, no. 3, pp. 165-186, Jul. 2001.

[8] G. Jung, D. Jin, and S. Park, "An efficient line based VLSI architecture for 2D lifting DWT," *in the 47th IEEE International Midwest Symposium on Circuits and Systems*, pp. 248-252, Jul. 2004.

[9] C. Lian, K. Chenm H. Chen, and L. Chen, "Lifting based discrete wavelet transform architecture for JPEG2000," *in IEEE ISCAS*, pp. 445-448, May. 2001.

[10] Y. Seo and D. Kim, "ASIC Design of Lifting Processor for Motion JPEG2000", *J. of KICS,* vol. 30, no. 5C, pp.344-354, May, 2005.

[11] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *IEEE Trans. Of Signal Process*, vol. 50, no. 4, pp. 966-977, Apr. 2002.

[12] H. Liao, M. Mandal, and B. Cockburn, "Efficient architectures for 1D and 2D lifting-based wavelet transform," IEEE Trans. Signal Process., vol. 52, no. 5, pp. 1315-1326, May. 2004.

[13] S. Barua, J. E. Carletta, K. A. Kotteri, and A. E. Bell, "An efficient architecture for lifting-based two-dimensional discrete wavelet transform," *Integr. VLSI J.*, vol. 38, no. 3, pp. 341-352, Jan. 2005.

[14] C. Xiong, J. Tian, and J. Liu, "Efficient architectures for two-dimensional discrete wavelet transform using lifting scheme," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 607-614, Mar. 2007.

[15] C. Cheng and K. K. Parhi, "High-speed VLSI implementation of 2D discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 393-403, Jan. 2008.

박 태 구 (Taegu Park)

2012년 2월 가톨릭대학교 정보통신전자공학부 졸업

<관심분야> 영상신호처리용 VLSI 설계

박 태 근 (Taegeun Park)

1985년 2월 연세대학교 전자공학과 학사

1988년 5월 미국 Syracuse University 컴퓨터공학과 석사

1993년 12월 미국 Syracuse University 컴퓨터공학과 박사

1998년 3월~현재 가톨릭대학교 정보통신전자공학부 교수

<관심분야> VLSI 설계, SoC 설계, CAD