논문 2004-41SD-8-11

다중필터 리프팅 방식을 이용한 고성능 라인기반 필터링 구조

(High-Performance Line-Based Filtering Architecture Using Multi-Filter Lifting Method)

서 영 호*, 김 동 욱*

(Young-Ho Seo and Dong-Wook Kim)

요 약

본 논문에서는 Motion JPEG2000 등의 이산 웨이블릿 기반의 고속 영상처리를 위해서 리프팅 방식의 효율적인 H/W 구조를 제안하였다. 리프팅 내부연산의 반복성을 이용하여 알고리즘 레벨에서 구조적인 사상을 적용하고 데이터 스케쥴링을 이용하여 최적화되고 간략화된 리프팅 기반의 필터링 셀의 구조를 제안한다. 이를 바탕으로 (9,7) 및 (5,3) 필터를 모두 수용할 수 있는 리프팅 커널의 구조를 구현하였다. 제안된 리프팅 커널은 일정 대기지연 시간 후에 연속적으로 데이터를 출력할 수 있는 간략화된 구조를 갖고 있다. 시간적인 순서로 입력되는 데이터에 대해서 일정한 출력을 발생할 수 있기 때문에 단순히 H/W를 추가하면 병렬적인 동작을 통해서 높은 출력율을 간단히 얻을 수 있다. 본 논문에서 제안된 리프팅 커널은 ASIC 및 FPGA 환경으로 모두 구현하였는데, ASIC으로는 삼성전자의 0.35μm CMOS 라이브러리를 이용하여 구현하였고 FPGA은 Altera사의 APEX을 타겟으로 하였다. ASIC의 경우 리프팅 연산을 위해 41,592개의 게이트 수와 라인 버퍼링을 위한 128Kbit의 메모리를 사용하였으며, FPGA의 경우 6,520개의 LE(Logic Element)와 128개의 ESB(Embedded System Block)을 사용하였다. 각각의 경우에 대해서 125MHz와 52MHz의 속도에서 안정적으로 동작할 수 있었다.

Abstract

In this paper, we proposed an efficient hardware architecture of line-based lifting algorithm for Motion JPEG2000. We proposed a new architecture of a lifting-based filtering cell which has an optimized and simplified structure. It was implemented in a hardware accommodating both (9,7) and (5,4) filter. Since the output rate is linearly proportional to the input rate, one can obtain the high throughput through parallel operation simply by adding the hardware units. It was implemented into both of ASIC and FPGA. The 0.35μm CMOS library from Samsung was used for ASIC and Altera was the target for FPGA. In ASIC, the proposed architecture used 41,592 gates for the lifting arithmetic and 128 Kbit memory. For FPGA, it used 6,520 LEs(Logic Elements) and 128 ESBs(Embedded System Blocks). The implementations were stably operated in the clock frequency of 128MHz and 52MHz, respectively.

Keywords: Lifting, Factorization, DWT, Line-based, Hardware, Filter

I. 서 론

현재 MPEG 및 JPEG을 이용한 영상 압축에서 주파수영역 변환 알고리즘으로 사용된 DCT(Discrete Cosine Transform)는 영상을 블록 단위로 처리하기 때문에 압축률이 높아짐에 따라서 블록화 현상(Blocking Effect)을 발생시키고 영상의 질을 떨어뜨렸다. 이에 비해 JPEG200 등에서 표준으로 채택한 웨이블릿 변환(Discrete Wavelet Transform, DWT)은 영상을 압축하는데 있어 기존의 DCT 방식에 비해 블록킹 현상이 나타나지 않고 각 부대역(Subband)별로 처리가 가능하여 압축률의 조절이 용이하여 고 압축률에서 상대적으로

* 종신회원, 광운대학교 전자재료공학과
(Dept. of Electronic Materials Eng., Kwangwoon University)

※ 본 논문은 정보통신부 정보통신연구진흥원에서 지원하고 있는 정보통신기초연구지원사업의 연구결과 (과제번호 : 04-기초-022)입니다.
접수일자: 2004년1월29일, 수정완료일; 2004년7월12일

좋은 화질을 나타낸다. 그러나 DWT가 타일링(Tiling)을 통해 분할된 영상을 대상으로 변환을 수행한다 할지라도 DCT 기반의 영상압축 방식에 비해 큰 영상을 변환단위로 사용하기 때문에 많은 메모리 용량과 메모리 접근횟수를 필요로 한다는 단점을 갖고 있다. 이를 해결하기 위한 알고리즘[1] 및 H/W의 구조적인 연구가 활발히 이루어지고 있다[2][3][4].

최근에는 훨씬 적은 계산량을 요구하는 리프팅(Lifting) 기법이 DWT를 위해 제안되었다. 리프팅을 이용한 변환은 컨벌루션(Convolution) 방식에 비해 계산량이 절반정도로 줄어 속도가 빠르고 메모리를 적게 사용하며, 정수 대 정수의 웨이블릿 변환이 용이하여 무손실 영상 압축에 유리하고 역변환을 쉽게 구현할 수 있다[5]. 이러한 이유로 H/W(Hardware)구현을 위한 접근[6][7][8]도 많이 이루어져 왔는데, H/W의 측면에서 리프팅 기법은 In-place 특성을 보유하고 있어서 컨벌루션-기반 필터링 기법에 비해 메모리양과 메모리 접근횟수의 감소를 가져오고 DWT와 역변환이 동일한 구조로 수행될 수 있는 장점을 가지고 있다.

지금까지 많은 H/W 구조들이 컨벌루션 기반의 DWT 연산을 위해 제안되어져 왔는데, 입력 데이터의 처리 방식에 따라 크게 직렬 구조(Serial architecture)와 병렬 구조(Parallel architecture)로 나눌 수 있다. 직렬 구조는 중간 데이터 저장소와 대기 지연시간을 줄이기 위해 각기 다른 DWT 레벨 계산을 인터리빙(Inter-leaving)하는 시스톨릭 배열(Systolic array)에 기반을 두면서 효율적인 필터 뱅크 구조를 위해 디지트 파이프라인(Digit pipeline)을 이용하였다[6-8]. 또한 병렬 구조는 인터리빙된 출력을 생성하고 각 레벨에 대해 파이프라인이 적용된다[9-11]. 최근에는 메모리 요구량과 계산량을 줄이는 리프팅-기반 DWT를 구현한 구조가 제안되고 있는데, [12]에서는 연산의 효율성을 위해 입력 영상을 블록으로 구분지어 처리하는 방식이 제안되었고, [13]에서는 (5,3)필터를 이용하여 데이터를 인터리빙하는 구조가 제안되었다. 또한 [14]에서는 EZW (Embedded zerotree wavelet) 알고리즘을 따르는 양자화 방식에 적합한 리프팅 구조가 제안되었고, [15]에서는 예측(Predict)과 갱신(Update)의 과정을 처리하는 단위 동작을 전체로 확장하는 구조가 연구되었다. [16]에서는 JPEG2000에서 규정하고 있는 여러 종류의 필터에 대해서 적용 가능하도록 확장성을 가진 연산단위를 제안하고 이들에 의한 리프팅 구조를 제안하였다. 그러나 [15]의 구조는 데이터패스부가 완벽히 파이프라인화되어 있지 않아서 임계경로가 큰 단점을 갖고 있다. 또한 [16]의 구조는 리프팅 연산을 그대로 H/W 구조화하여 구조적인 간결함을 갖고 있지만 연속된 입력 데이터에 대한 처리결과가 모호성을 가지고 있고 리프팅 연산과 내부 메모리와의 관계가 명확히 규명되지 않았다. [17]에서는 파이프라인된 리프팅 연산과 확장성을 가진 구조를 제안하여 고속의 성능을 보여주고 있지만 리프팅 알고리즘을 그대로 사상하여 파이프라인을 위해 많은 수의 레지스터가 요구되고 라인 기반의 리프팅 방식을 명확히 나타내지 못하는 단점을 가진다. 또한 JPEG2000이 JPEG을 대체할 것으로 예상되고 있어 다양한 상용 IP(Intellectual Property)들이 개발되고 있다[18][19][20][21][22].

본 논문에서는 JPEG2000 등의 이산 웨이블릿 기반의 고속 영상처리를 위한 효율적인 리프팅 구조를 제안한다. 시간 순서에 따라서 입력되는 데이터에 의존하여 리프팅 연산 순서를 재조정하고 이를 바탕으로 셀(Cell) 기반의 리프팅 연산 구조를 제안한다. JPEG2000의 손실압축과 무손실 압축과정을 모두 수용할 수 있도록 (5,3)과 (9,7) 필터에 대한 동작을 모두 포함하는 구조를 가지도록 한다. 이 경우 연산량의 차이에 따라서 (5,3) 필터의 경우 (9,7) 필터의 두 배에 해당하는 데이터율을 가진다. 그리고 이전 연구들에서 JPEG2000을 목적으로 하여 다양한 H/W 리프팅 구조를 제시하고 있으나 동작의 복잡성이나 여러 부가회로의 부재 및 메모리와의 관계 제시의 부족 등의 이유로 실제적인 동작 및 구현에 어려움이 많고 상용화를 위한 IP로서의 가치가 부족한 것이 사실이다. 따라서 본 논문에서는 그러한 점들을 보완하고자 H/W 구조와 제어의 단순화에 중점을 두었다.

본 논문은 다음과 같이 구성된다. 먼저 Ⅱ장에서는 본 논문에서 사용하고 있는 리프팅 연산에 대해서 간단히 설명한다. 그리고 Ⅲ장에서는 제안된 구조를 설명하고 Ⅳ장에서 그 구조에 대한 구현 및 결과를 설명한다. 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. 리프팅 알고리즘

본 장에서는 기본적인 리프팅 알고리즘에 대해서 설명하고 리프팅 방식으로 DWT를 수행하는 H/W의 내부적인 수체계를 리프팅 연산의 정밀도 분석 실험을 통

해 결정한다.

1. 리프팅 알고리즘

리프팅 방식의 기본적인 원리는 웨이블릿 필터의 다상 행렬(Polyphase matrix)을 삼각 행렬(Triangular matrix)과 대각 행렬(Diagonal matrix)로 인수분해(factoring)하는 것이다[23-24]. 이는 밴드 행렬(banded -matrix) 곱셈에 의해서 웨이블릿이 수행되도록 하는 것이다. $\tilde{h}(z)$와 $\tilde{g}(z)$를 각각 저주파 및 고주파 해석 필터(Analysis filter)라 하고 $h(z)$와 $g(z)$를 합성 필터(Synthesis filter)라 할 때 아래와 같이 식 (1)과 (2)로 각각의 다상 함수를 정의할 수 있다.

$$\tilde{P}(z) = \begin{matrix} \tilde{h_e}(z) & \tilde{h_o}(z) \\ \tilde{g_e}(z) & \tilde{g_o}(z) \end{matrix} \quad (1)$$

$$P(z) = \begin{matrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{matrix} \quad (2)$$

$(\tilde{h}, \tilde{g})$가 상보적인 필터쌍이라면 $\tilde{P}(z)$는 반드시 아래의 식 (3) 혹은 (4)와 같은 리프팅 과정들로 인수분해 될 수 있다.

$$\tilde{P_1}(z) = \begin{matrix} K & 0 \\ 0 & \frac{1}{K} \end{matrix} \prod_{i=1}^{m} \begin{matrix} 1 & \tilde{s_i}(z) \\ 0 & 1 \end{matrix} \begin{matrix} 1 & 0 \\ \tilde{t_i}(z) & 1 \end{matrix} \quad (3)$$

$$\tilde{P_2}(z) = \begin{matrix} K & 0 \\ 0 & \frac{1}{K} \end{matrix} \prod_{i=1}^{m} \begin{matrix} 1 & 0 \\ \tilde{t_i}(z) & 1 \end{matrix} \begin{matrix} 1 & \tilde{s_i}(z) \\ 0 & 1 \end{matrix} \quad (4)$$

여기서 K는 상수이다. 그림 1은 식 (3)의 $\tilde{P_1}(z)$에 해당하는 일반적인 리프팅 과정을 나타내고 있는데, 리프팅을 이용한 웨이블릿 변환은 아래와 같이 크게 네 단계로 구성된다.

가. 분할(Split) 및 결합(Merge)

입력 신호 $x[n]$을 짝수 번째 신호 $x_e[n]$과 홀수 번째 신호 $x_o[n]$의 두 성분으로 분할(정변환 리프팅) 및 결합(역변환 리프팅)한다.

나. 예측(Predict)

예측 연산자 $\tilde{t}(z)$를 이용하여 짝수 번째 신호로부터 홀수 번째 신호를 예측할 때 얻어지는 에러에 해당하는 웨이블릿 계수를 계산한다. 예측 과정을 통해 얻어진 결과는 웨이블릿 변환에서 고주파 필터링의 결과에 해당한다.

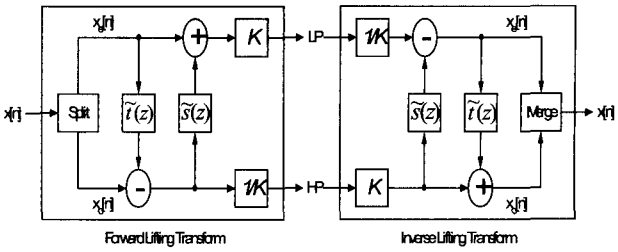

그림 1. 팩토링을 이용한 정방향/역방향 리프팅 방식의 구조도

Fig. 1. Forward/inverse lifting scheme using factoring.

다. 갱신(Update)

입력 신호 $x[n]$을 근사하여 나타내는 스케일링(Scaling) 계수를 얻기 위해 $x_e[n]$과 예측 오차를 결합한다. 이것은 웨이블릿 계수에 갱신 연산자 $\tilde{s}(z)$를 적용한 다음 $x_e[n]$를 더하여 구한다. 갱신 과정을 통해 계산된 결과는 웨이블릿 변환에서 저주파 필터링의 결과에 해당한다.

라. 조정(Scaling)

예측과 갱신 결과는 일정한 상수 K 또는 1/K에 의해서 크기가 조정되고 이 결과가 리프팅을 이용한 웨이블릿 변환의 최종 결과에 해당한다.

정방향 리프팅과 역방향 리프팅은 서로 상보적인 연산 순서를 가지고 있는데, 그림 1과 같이 연산 순서가 (예측—갱신)으로 진행된다면 역변환 시에는 (갱신—예측)의 순서를 따른다. 그림 1과 달리 식 (4)에 의한 리프팅 방식을 적용할 경우에도 마찬가지로 정변환 리프팅의 경우에 (갱신—예측)의 순서로 연산이 진행되었다면 역변환 리프팅은 (예측—갱신)의 순서대로 연산을 진행한다. 그리고 리프팅은 선형성을 가지기 때문에 입력 데이터가 정수일 경우 연산결과도 정수형을 유지할 수 있어서 가역적인 정수형 리프팅 연산이 가능하다.

리프팅의 구체적인 연산방식 및 예시는 2절에서 H/W 구조를 위한 연산방식 분석과 함께 설명하도록 한다.

2. 고정소수점의 정밀도 분석

신호처리 기반의 알고리즘들은 C 언어와 같은 상위 레벨 언어를 이용하여 개발되고 검증되는 것이 일반적인 방법이다. 그러나 컴퓨터 환경에서 프로그래밍 언어가 갖는 정밀도는 상당히 뛰어난 수준으로 H/W 기반의 칩 환경에서 동일한 결과를 얻기 위해서는 많은 데

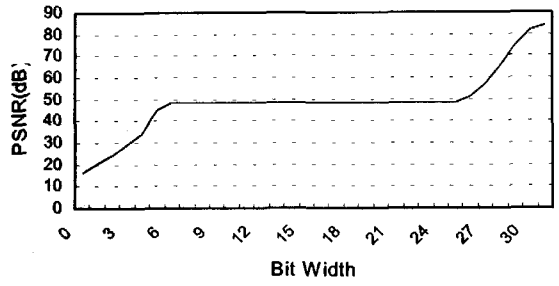

그림 2. 필터계수의 정밀도 분석
Fig. 2. Precision analysis of filter coefficient.

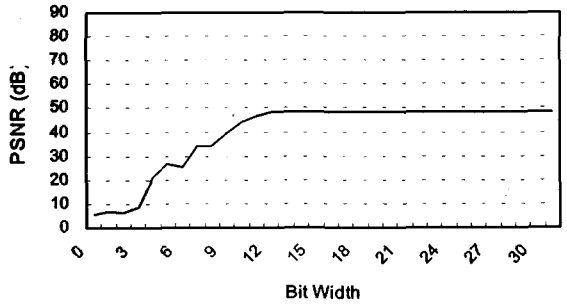

그림 3. 웨이블릿 계수의 정밀도 분석
Fig. 3. Precision analysis of wavelet coefficient.

이터량과 연산량이 필요하다. 즉, H/W의 성능과 비용 등의 제약이 고려된 합리적인 정밀도 분석은 필수적인 과정으로 인식될 수 있다. 따라서 정밀도와 유연성이 높은 소프트웨어 기반의 상위레벨 언어를 이용하여 알고리즘과 구조를 검증할 때 중간 연산결과들에 대해 H/W와 동일한 고정소수점 제약을 부가한다. 또한 이러한 환경을 통해 알고리즘과 구조를 검증하여 개발된 알고리즘이 실제 H/W에서 어떤 성능을 가질 수 있는지 정확히 예측 및 분석할 수 있고, 알고리즘의 개발 단계에서 H/W의 성능을 고려할 수 있어 통합된 형태의 개발 환경을 구성할 수 있다. 본 연구에서는 연산 결과에 영향을 미치는 필터 계수와 웨이블릿 계수를 고정소수점 시뮬레이션을 통해서 비트 제약이 성능에 미치는 영향을 분석하였고, 이를 바탕으로 H/W 구현을 위한 수체계를 확립하였다. 그 결과를 그림 2와 3에 각각 나타내었는데 그림 2에는 필터계수, 그림 3에서는 웨이블릿 계수의 고정소수점 시뮬레이션을 통한 PSNR 결과를 각각 나타내었다. 실험 환경은 C++ 언어를 사용하였고 500개의 정지영상을 이용하였다. 또한 무손실 압축을 위해 사용되는 (5,3)필터는 실험에서 제외되었고 손실압축을 위한 (9,7)필터만이 사용하였다. 그림 2에 보이는 것과 같이 필터계수의 경우 소수점 이하 비트가 8비트 이상이면 약 50dB 정도로 손실압축에 아무런 문제가 없을 정도의 성능을 보이고, 30비트 이상이면 90dB 정도로 거의 완전 복원이 가능하다. 그림 3에서는 웨이블릿 계수가 소수점 이하 13비트 이상으로 표현된다면 50dB 이상의 좋은 성능을 나타낸다는 것을 보이고 있다. 그리고 그 이상의 정밀도를 부가한다고 해도 결과에는 그다지 영향을 주지 않음을 알 수 있다.

III. 제안된 리프팅 커널의 구조

1. 리프팅과 인과시스템의 구성

리프팅 연산이 수행되는 절차를 도식적으로 나타내면 그림 4와 같다. 그림 4에서는 정방향 리프팅 연산을 거친 후 저주파 영역(Low-Pass Filtering Part)과 고주파 영역(High-Pass Filtering Part)으로 주파수 변환하는 절차를 보이고 있다. 또한 리프팅을 이용하여 웨이블릿 계수로 변환된 데이터는 재정렬 과정을 거친 후에 동일한 구조의 연산절차를 거쳐서 원래의 데이터로 복원된다. 그림 4에서는 (9,7) 필터에 대한 리프팅 연산을 나타내고 있는데 (5,3) 필터에 대해서는 연산 구조의 절반만을 사용한다. 그림 4에서 보이는 것과 같이 (9,7) 필터를 사용할 경우 연산 후에 출력되는 데이터의 위치를 기준으로 양 옆 4개(NTSC와 PAL의 모든 경우를 고려)의 데이터가 필요하다. 원래의 데이터(Original Data)가 시간적인 순서대로 입력된다고 할 때 리프팅 연산방식은 비인과적(Non-causal)인 시스템에 해당되고 이는 부가적인 알고리즘의 수정이나 H/W적인 버퍼링이 필요하다. 따라서 H/W로 구현할 경우에 복잡도를 증가시키고 효율성을 떨어뜨린다.

그림 4에 나타낸 리프팅 연산구조에 인과성을 부여한 후 그대로 H/W로 사상하면 그림 5와 같은 H/W로 구성할 수 있다. 그림 5의 구조에서 처음 웨이블릿 변환된 계수를 구하기 위해서는 최소 10 클록의 대기지연시간이 필요하고 그 후부터는 연속적으로 계수가 출력된다. 그러나 그림 5에 나타난 구조는 입력 데이터 길이만큼의 H/W가 요구되기 때문에 실제 구현을 위해서는 적합하지 않다.

그림 4의 리프팅 연산구조와 그림 5의 H/W 구조를 보면 동일한 연산 구조가 반복되고 있는 것을 볼 수 있다. 따라서 이들 반복적인 구조에 대해서 시간적인 순서와 동작의 유사성을 고려하여 시간축에 대해 H/W

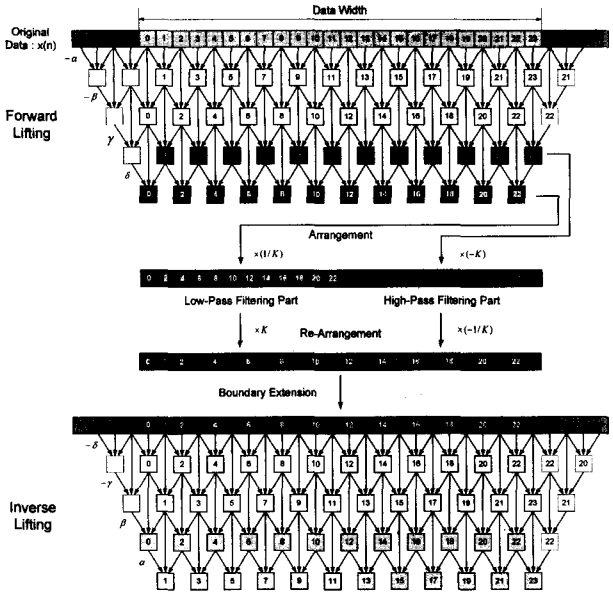

그림 4. 정방향/역방향 리프팅의 상세 구조
Fig. 4. Detailed forward/inverse lifting structure.

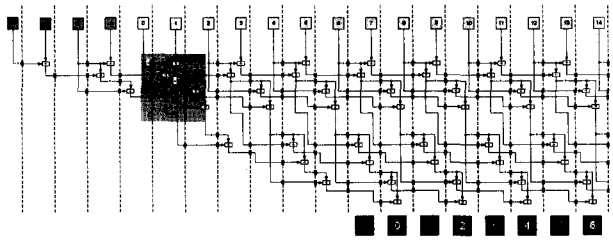

그림 5. 연산의 재배열을 통한 인과적 리프팅 구조
Fig. 5. Casual lifting structure by re-arrangement of calculation.

구조 사상(Hardware architecture projection)을 적용하고 동일한 동작의 연산블록으로 최적화시키는 것이 가능하다. 그림 5에서 회색으로 표시된 부분이 전체를 구성할 수 있는 단위 블록(셀, cell)에 해당한다. 다음 장에서 리프팅 연산 구조로부터 단위 블록을 이끌어 내도록 한다.

2. 제안된 LBFC의 구조

본 절에서는 앞 절에서 살펴본 리프팅 연산 블록의 구조사상 과정을 연산 단계에서부터 최적화시키도록 한다. 그림 6에 이 사상과정을 도식적으로 나타내었다. 그림 6의 윗부분은 그림 4의 정방향 리프팅에 해당하는데, 그림에서 나타냈듯이 (9,7) 필터를 이용한 리프팅은 리프팅 계수만 다르고 방식은 동일한 총 4단계의 곱셈과 덧셈과정을 거친다. 또한 시간적인 순서로 볼 때 단계가 동일한 순간에 수행되지 않기 때문에 적절한 데이터 버퍼링과 스케쥴링을 적용한다면 그림 6의 오른쪽

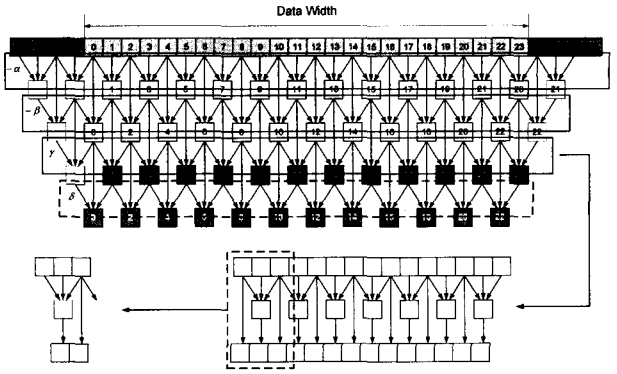

그림 6. 알고리즘의 구조사상을 통한 단위연산 구조
Fig. 6. Unit calculation structure by structure mapping of algorithm.

아래와 같은 하나의 단계로 대치시킬 수 있다. 마지막으로 하나의 단계로 사상된 리프팅 연산은 시간축에서 볼 때 동일한 구조를 가지고 동일한 연산을 수행하므로 그림 6의 아래쪽 왼편 그림과 같은 하나의 연산 블록으로 사상할 수 있다. 물론 이 경우 연속적으로 입력되는 데이터를 처리할 수 있어야 하는 기본사항을 만족시켜야 한다.

(5,3) 필터와 (9,7) 필터를 위한 동작을 모두 만족하면서 연속적으로 입력되는 데이터를 처리할 수 있는 리프팅을 위한 필터링 셀의 구조를 제안하고 이를 그림 7에 나타냈다. 본 논문에서 제안된 H/W를 리프팅 기반의 필터링 셀(LBFC, Lifting-Based Filtering Cell)이라고 한다. 그림 7에 나타낸 LBFC가 그림 6의 왼쪽 아래의 연산 구조를 그대로 나타내는 것은 아니며, 출력 시 일정 대기지연 시간을 가지므로 출력 순서는 조금 다르다.

본 논문에서는 수직방향의 처리 위해서는 라인 기반의 필터링을 채택하고 있으므로 그림 7의 (a)에서 각 레지스터들이 그림 7의 (b)에서는 라인 버퍼(Line buffer)로 대체된다.

그림 7의 LBFC에서 각 레지스터들은 데이터의 입력과 연산된 데이터의 저장 및 지연을 위한 버퍼링을 담당한다. 왼쪽의 쉬프트 레지스터는 필터 계수를 저장하는데, 필터링 동작의 초기에 해당 필터 계수로 프로그래밍되고 적절한 시간순서에 따라 필터링을 위한 필터 계수를 출력한다. 클록순서에 따라서 레지스터들의 내용이 어떻게 바뀌고 출력은 어떤 순서를 가지는지를 표 1에 나타냈다. 5 클록의 대기지연 시간 후부터 그림 4에 나타낸 연산 순서대로 올바른 결과가 출력되는 것(Output)을 볼 수 있다. 본 논문에서 제안된 구조는 곱

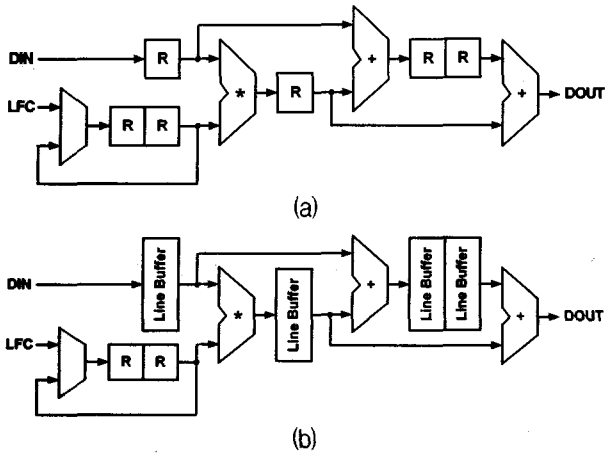

그림 7. 리프팅을 위한 LBFC의 설계; (a) 수평방향 LBFC, (b) 수직방향 LBFC
Fig. 7. LBFC architecture for lifting; (a) horizontal LBFC, (c) vertical LBFC.

표 1. LBFC의 동작
Table 1. Operation of LBFC.

Clock	Input	R1	R2	R3	R4	Output
1	4	0	0	0	0	0
2	3	20	0	0	0	0
3	2	3	4a	20	0	0
4	1	2	0	3+4a	20	0
5	0	1	2a	2	3+4a	20
6	1	0	0	1+2a	2	2a+3+4a
7	2	1	0a	0	1+2a	2
8	3	2	0	1+0a	0	0a+1+2a
9	4	3	2a	2	1+0a	0
10	5	4	0	3+2a	2	2a+1+0a
11	6	5	4a	4	3+2a	2
12	7	6	0	5+4a	4	4a+3+2a

셈기의 지연만큼 임계경로를 가지게 되고 경계처리와 무관한 구조이다. 그림 4, 5, 그리고 표 1에서는 JPEG2000 표준안에 맞추어 대칭적인 경계 확장방식을 따르고 있다.

그림 7과 표 1에서 볼 수 있듯이 제안된 리프팅 연산 H/W에서 입력된 데이터는 최소의 생존시간(Lift time)을 가지면서 해당 데이터가 요구되는 연산에 오직 단 한 번만 관여하여 데이터 사용에 대한 효율성이 최대화 되어 있다. 뿐만 아니라 리프팅의 특성상 하나의 중간 연산결과가 여러 최종 연산결과에 영향을 미치는데, 표 1에서 보이듯이 한 번 연산된 중간 결과들은 파이프라인화된 귀환 데이터 경로를 따라서 적절히 여러 연산에 관여함으로써 두 번 이상 연산되지 않고 결국 최소의 연산수를 가지게 되어 극대화된 성능을 보인다.

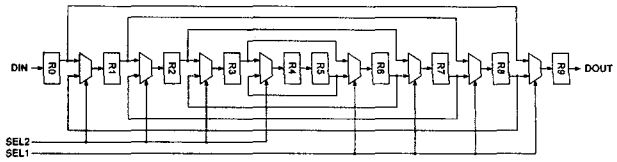

그림 8. BPU의 구조
Fig. 8. Architecture of BPU.

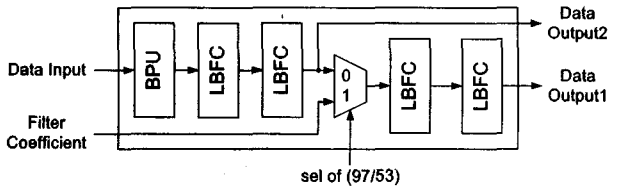

그림 9. LBFC를 이용한 LFDWT 구조
Fig. 9. LFDWT architecture using LBFC.

3. 제안된 경계 처리 블록의 구조

JPEG2000 표준안에서는 영상의 경계에서 대칭적으로 확장하는 방식을 채택하고 있는데, 이를 위한 H/W 구조(BPU, Boundary Processing Unit)를 그림 8에 나타냈다. LBFC와는 독립적으로 동작이 가능하도록 하였고 입력되는 직렬 데이터에 대해서 간단한 버퍼링 동작을 수행하여 앞부분과 뒷부분에 4개의 데이터를 덧붙임으로써 대칭적으로 데이터를 확장한다. 그림 8에서 R4와 R5 레지스터를 기점으로 MUX를 통해서 서로 데이터를 대칭적으로 전달할 수 있는 것을 볼 수 있다.

4. 제안된 리프팅 커널의 구조

그림 9는 그림 7의 LBFC를 이용한 리프팅 방식의 필터링 구조인 LFDWT(Lifting Filter for DWT)를 나타내고 있다. (9,7) 필터를 이용하는 경우 필터계수와 4단계의 연산을 거치면서 리프팅 연산이 수행되기 때문에 4개의 LBFC가 요구되고 20 클록의 대기 지연을 가진다. (5,3) 필터를 이용하는 경우는 2개의 LBFC만 필요로 한다. 또한 (5,3) 필터를 사용할 경우 (9,7) 필터에 비해서 1/2의 H/W 자원만 사용하여 두 배의 필터링 율을 얻을 수 있는데, 그림 9에서 앞의 두 LBFC와 MUX 다음의 두 LBFC에 의해서 병렬적인 필터링을 수행한다. 이 때 MUX에 의해서 3번째 LBFC로 직접 입력을 넣어준다.

그림 10에는 그림 9의 LFDWT를 이용하여 JPEG 2000에서 사용될 수 있는 리프팅 커널의 구조를 나타내었다. 리프팅을 이용한 필터링은 레벨단위로 이루어지

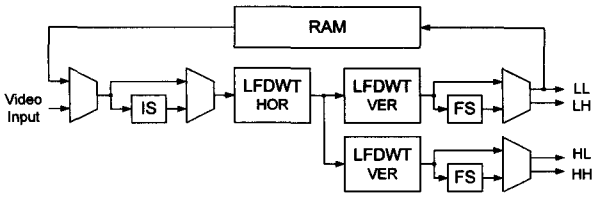

그림 10. LFDWT를 이용한 리프팅 커널의 구조
Fig. 10. Lifting kernel architecture using LFDWT.

고 f의 속도로 입력된 Data Input은 연산을 거친 후 f/4의 속도로 네 개의 출력을 병렬로 출력하여 결국 입력과 동일한 f의 출력율을 가진다. 출력 중에서 LL은 다음 레벨 연산을 위해 RAM에 저장된다. 그림 10에서 IS와 FS는 그림 4의 K값에 의한 역방향 및 순방향 스케일링 블록을 각각 나타낸다.

Ⅳ. 실 험

본 논문에서 제안된 리프팅 커널은 ASIC 및 FPGA 환경으로 모두 구현하였다. ASIC은 삼성전자의 0.35μm CMOS 라이브러리를 이용하여 구현하였고 FPGA은 Altera사의 APEX를 타겟으로 하였다. 리프팅 알고리즘은 C++언어를 이용하여 검증하였고 이를 Verilog-HDL을 사용하여 RTL 수준의 H/W로 사상하였다. 그리고 H/W에 대한 동작을 용이하게 검증하기 위해서 Verilog 2001의 문법을 일부 사용하였고 이때 ModelSim (5.7버전 이상에서 지원)의 시뮬레이터를 이용하였다. Verilog 2001로 구현된 H/W는 ASIC과 FPGA로 구현 시 해당 라이브러리와 지원되는 매크로로 교체하였다. 제안된 리프팅 커널은 JPEG2000 표준을 위한 H/W이므로 입력 영상을 타일 단위로 처리한다. 따라서 라인버퍼는 최대 512×512 크기의 타일을 수용할 수 있도록 하였고 ASIC의 경우에는 제공되는 메모리 컴파일러를 통한 임베디드 RAM을 사용하였다. 또한 FPGA의 경우에는 해당 FPGA 내부의 임베디드 시스템 메모리들을 사용하였다. 3장에서 언급한 것과 같이 구현된 H/W는 (9,7) 및 (5,3) 필터를 모두 수용할 수 있고 입력되는 데이터의 양과 동일한 출력율을 가진다.

1. 구현 및 동작 검증

표 2에는 Lena 영상에 대한 리프팅 연산의 실제적인 예를 나타내고 있다. 표에서 Step은 그림 4와 6에서 열방향의 단위 연산을 나타내는 것으로, 각 단위 연산은 다음의 계수에 의해서 수행된다[23].

표 2. Lena 영상에 대한 리프팅 연산 예
Table 2. Example of lifting operation for Lena image.

#	Data	Lifting				Binary of Step4
		Step 1	Step 2	Step 3	Step 4	
4	171					
3	163	-379.458				
2	171	171	211.887			
1	193	-392.284	-379.458	6.310		
0	198	198	239.566	239.566	245.163	011110101_0010100
1	193	-392.284	-392.284	6.310	6.310	000000110_0100111
2	171	171	211.887	211.887	212.068	011010100_0001000
3	163	-379.458	-392.284	-5.902	-5.902	111111010_0001100
4	171	171	211.209	211.209	210.95	011010010_1111001
5	182	-379.492	-379.458	5.318	5.318	000000101_0101000
6	183	183	224.632	224.632	229.406	011100101_0110100

$$\text{Step 1}: \alpha = -1.586134342$$
$$\text{Step 2}: \beta = -0.052980118$$
$$\text{Step 3}: \gamma = 0.8829110762$$
$$\text{Step 4}: \delta = 0.4435068522$$

그림 11에 수평방향의 LBFC를 동작시킨 시뮬레이션 결과를 보이고 있는데 이는 표 2의 Step 1의 연산 결과에 해당하는 것으로 출력인 "dout"의 값이 '1010000100.1000100(-379.458)', '0010101011.0000000(171)', '1001110111.1011011(-392.284)', '0011000110.0000000(198)'의 순서로 정확하게 출력되는 것을 볼 수 있다. 또한 그림 12은 한 레벨에 대한 수평방향 리프팅 필터링의 최종 결과에 해당하는 것으로 "dout1"신호의 값이 표 2의 Step 4의 결과와 동일함을 볼 수 있다. 물론 표 2는 C++ 언어를 이용한 64비트 floating 연산을 수행한 것이기 때문에 그림 11 및 12의 고정소수점 연산과 소수점 하위 비트에서 약간의 오차를 보일 수 있다. 그러나 이러한 오차는 JPEG2000의 영상 압축 과정에서 발생하는 양자화 및 패킷 분할 전송 과정에서 발생하는 오차에 포함되어 실제적으로 결과에 영향을 주지는 않는다.

또한 그림 13는 수직 방향 LBFC에 대한 결과를 나타내고 있다. 수직 방향의 경우에 라인 기반의 DWT를 수행하게 되는데, 동작을 용이하게 관찰할 수 있도록 행방향으로 동일한 값들이 존재하는 영상을 가정하여 시뮬레이션을 수행하였다. 따라서 그림 13에서 보이듯이 4 클록 동안 동일한 출력이 발생하는데, 이는 관찰의 용이성을 위해서 동일한 값을 가진 4 열의 영상을 입력한 결과에 해당한다.

그림 11, 12을 통해서 직관적으로 알 수 있듯이

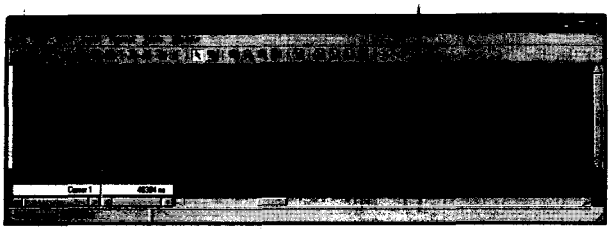

그림 11. 수평방향 LBFC의 시뮬레이션 결과
Fig. 11. Simulation result of horizontal LBFC.

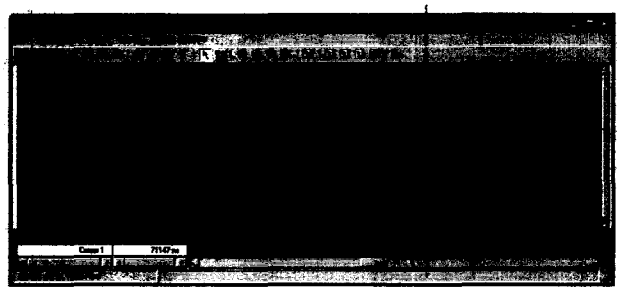

그림 12. 수평방향 LFDWT의 시뮬레이션 결과
Fig. 12. Simulation result of vertical LFDWT.

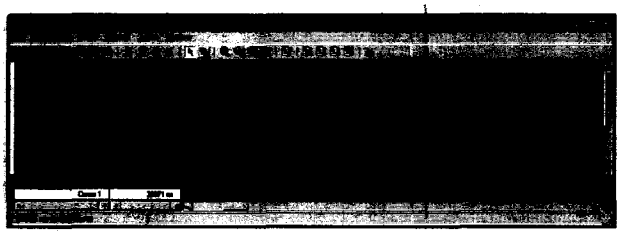

그림 13. 수직방향 LBFC의 시뮬레이션 결과
Fig. 13. Simulation result of vertical LBFC.

LBFC는 복잡한 제어가 필요하지 않고 단순히 실제 영상의 순서대로 화소 혹은 계수값을 입력만하면 출력을 발생하는 매우 단순한 동작 및 제어관계를 보이고 있다. 또한 동작의 초기에 "**fc**"와 "**load**"를 이용하여 필터 계수를 프로그래밍할 수 있게 하여 다양한 필터에 대해서 적용 가능한 구조를 가지고 있고 각 필터의 특성에 따라서 LBFC를 선형적으로 확장시켜주면 원하는 동작을 얻을 수 있다. 이러한 H/W의 특성에 따라서 수직 LFDWT 동작도 수직 LBFC의 단순한 확장으로 구현이 가능하다(이에 대한 동작의 시뮬레이션 결과는 생략).

2. ASIC 및 FPGA 구현과 성능 비교

이미 설명한 것과 같이 제안된 리프팅 커널은 ASIC과 FPGA로 모두 구현하였다. 구체적인 결과와 다른 연구 및 상용 제품과의 비교를 표 3과 표 4에 나타내었다. [11]에서 제안된 것을 제외하고는 모두 리프팅 알고리즘을 사용하였다. 일반적으로 알려진 것과 같이 컨벌루션 방식이 H/W양이나 메모리양이 많은 것을 볼 수 있다.뿐만 아니라 가시적으로 나타내지는 않았지만 실제 구현에서 가장 중요한 요소인 메모리 접근횟수도 리프팅 방식에 비해 컨벌루션 방식이 상당히 많은 것이 일반적이다. H/W양은 [16]에서 제안한 것이 가장 작은데, 이는 (5,3) 필터만이 수용가능하기 때문으로 (5,3) 필터를 이용한 필터링이 (9,7) 필터를 이용한 필터링에 비해서 리프팅의 경우에 약 2배 이상의 복잡도를 가지는 것을 고려하여 다른 H/W와 동일한 기준을 적용하면 약 2배정도로 H/W양이 증가해야 한다. 두 필터를 수용하면서 H/W 양이 가장 작은 것이 [17]에서 제안된 H/W인데, 출력의 데이터율이 다른 것들에 비해서 1/2인 단점이 있다. [17]에서 제안된 구조는 데이터율을 높이기 위해서 수직 필터링 블록을 하나 증가시켜야 하는데, 이 경우에 곱셈기(*)와 덧셈기(+)가 원래의 H/W에서 약 30%정도 증가해야 한다. 또한 내부적인 파이프라인 단계를 맞추기 위한 레지스터와 FIFO가 상당히 많이 내장되어야 하는 단점이 있다. 그리고 제어의 복잡도를 살펴보면 [11]과 본 논문의 H/W가 비교적 단순한 제어 특성을 보이고 [16]과 [17]은 상당히 복잡한 제어 관계를 가진다.

표 3. 제안된 H/W의 자원 사용
Table 3. Resource usage of the proposed hardware.

Arch	Filter	Image Size	*	+	Storage Size	Output Rate	Operation Frequency	Gate Count	Control Complexity
[11]	(9,7)/ (5,3)	N×N	36	36	$N^2/4+KN+K$	f	-	-	Simple
[16]	(5,3)	129× 129	4	8	236 Kbits	f	200Mhz (0.18um)	15,000 (0.18um)	Complex
[17]	(9,7)/ (5,3)	256× 256	8	16	57.6Kbit	$f/2$	110Mhz (FPGA)	2,363 SL (+3,449 Reg)	Complex
Ours	(9,7)/ (5,3)	512× 512	12	24	128Kbit	f	125Mhz (0.35um)	41,592 (0.35um)	Simple

필터의 적응성, 처리할 수 있는 영상크기에 따른 저장 공간(메모리의 양), 동일 성능에 대한 곱셈기 및 덧셈기의 수, 그리고 제어의 복잡도 등을 비교할 때 본 논문에서 제안된 리프팅 커널이 가장 우수한 것을 볼 수 있다.

JPEG2000의 저변이 확대되면서 업계에서 상용화된 다양한 리프팅 커널을 출시하고 있는데, 그러한 IP들과 본 논문에서 제안하고 구현한 H/W를 표 4에서 비교하였다. 표 4에서 보인 IP들은 JPEG2000을 위한 것이므로 표 3의 연구들과 달리 입력 영상을 타일 단위로 규정하고 있고 제품으로 이미 판매하는 것들이기 때문에 실제로 구현이 가능한 것들로 그 성능 또한 명확히 나타내고 있다.

표 4. 상용 IP들과 성능 비교
Table 4. Performance comparison with commercial IPs.

Design	Tile Size	Transform	Filter	Target Technology	Gate Count	Clock (MHz)	Memory
BA113FDWT[18]	128×128	F	(9,7)(5,3)	Altera EP1S2	4,291 LE	130	2 MRAM
				Xilinx XC2V100	3,381 SL	135	42 RAMB16
				ASIC(0.18um)	50,000 gate	205	112Kbits+512K bits(tile buffer)
RC_2DDWT[19]	512×512	F/I	(9,7)	Altera	3,280 LE	65	External RAM
				Xilinx Vertex II	1,698 SL	95	External RAM
LB-2DFDWT[20]	256×256	F	(9,7)	Xilinx Vertex II	2,227 SL	51	14 Block RAM
BB-2DFDWT[21]	-	F	(5,3)	Xilinx Vertex II	946 SL	52	10 Block RAM
CS6210 [22]	128×128	F	(9,7)(5,3)	Altera APEX20	7,381 LE	47	24 ESB
				Xilinx VirtexE-8	3,784 SL	55	24 BRAB
				ASIC(0.18um)	55,000 gate	150	50KB
Ours	512×512	F/I	(9,7)(5,3)	Altera APEX20K	6520 LE	52	128 ESB
				ASIC(0.35um)	41,592 gate	125	128Kbit

표 4에서 'Tile Size'는 입력 영상크기를 뜻하고 이는 'Memory'의 크기와 직결된다. 'Transform'은 순방향 및 역방향 변환을 수행할 수 있는지를 나타내는 것이고 'Filter'는 어떤 필터를 지원할 수 있는지를 나타낸다. 그리고 구현 환경(Target Technology)과 게이트 수(Gate Size), 동작 속도(Clock) 등을 비교하고 있다.

먼저 [21]의 경우 (5,3) 필터만을 지원하면서 순방향 필터링만 가능한 Soft IP 형태로서 성능이 가능 떨어지는 것을 볼 수 있다. 또한 [19] 및 [20]도 [18]과 [22]에 비해서 성능이 낮은 적용분야에 사용되는 IP이다. [18]과 [22] 및 제안된 H/W를 비교하면 H/W 양에 비해서 순방향 필터링만 수행하는 [18]이 가장 많은 자원을 사용하고 본 논문에서 제안된 H/W가 순방향 및 역방향 필터링을 모두 지원하면서 ASIC의 경우 41,592개로 가장 작고 FPGA의 경우 6520개의 LE(Logic Element)로 두 번째로 작은 자원을 사용하고 있다. 동작 속도의 경우에 [18]과 [22]가 제안된 H/W에 비해 높은 것을 볼 수 있으나, 이는 공정 및 타겟 디바이스의 변경으로 극복될 수 있는 부분이라 판단된다. 또한 메모리의 사용을 보면 동일한 타일 크기를 지원한다고 가정할 경우에 [18]은 112Kbit×4이고 [19]는 50K×8×4의 메모리를 사용하여 128Kbit를 사용하는 본 논문의 H/W가 가장 작게 사용한다.

V. 결 론

본 논문에서는 JPEG2000 등의 이산 웨이블릿 변환 기반의 고속 영상처리를 위한 리프팅 구조를 제안하고 그에 기반하여 필터링 H/W를 효율적으로 구현하였다. 시간 순서에 따라서 입력되는 데이터에 의존하여 리프팅 연산 순서를 재조정하고 이를 바탕으로 셀(Cell)기반의 리프팅 연산 구조를 제안하였다. 그리고 JPEG2000에서 규정한 것과 같이 손실압축과 무손실 압축과정을 모두 수용할 수 있도록 (5,3)과 (9,7) 필터 모두를 사용할 수 있도록 하였다. 실제적인 동작 및 구현이 용이하도록 H/W 형태와 제어를 단순화시켰고 셀 기반의 일정한 H/W 구조는 단순한 H/W의 추가를 통해 출력율의 증가를 얻을 수 있었다. 제안된 리프팅 커널은 ASIC 및 FPGA 환경으로 모두 구현하였는데 기존의 연구들과 현재 시판되고 있는 상용화 IP들과의 비교를 통해서 그 우수성을 검증하였다.

JPEG2000을 비롯하여 웨이블릿 변환 기반 제품의 저변이 확대되고 있고 모든 영상관련 시스템에 대한 보안성이 대두대고 있는 시점에서 구현된 H/W는 독립적으로 하나의 영상처리 제품이 될 수 있고 SOC(system-on-a-chip)를 구성하는 데도 다양하게 이용될 수 있을 것으로 사료된다.

참 고 문 헌

[1] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," IEEE Signal Processi-ng Magazine, pp. 14-38, 1991.

[2] M. Ravasi, et al., "Wavelet Image Compression for Mobile/Portable Applications", IEEE Trans. on Consumer Electronics, Vol. 45, No. 3, pp. 794-803, August 1999.

[3] A . Grezeszczak, et al., "VLSI Implementation of Discrete Wavelet Transform", IEEE Trans. on VLSI Systems, Vol. 4, No. 4, pp. 421-433, 1996.

[4] Po-Cheng Wu and Liang-Gee Chen, "An Efficie-nt Architecture for Two-Dimensional Discrete Wavelet Transform", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 4, pp. 536-545, 2001.

[5] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," J. Appl. Comp. Harm. Anal., vol. 3, no. 2, pp. 186-200, 1996.

[6] W. Jiang and A. Ortega, "Lifting factorization-based discrete wavelet transform architecture design," IEEE Trans. Circuits Syst. Video Technol., vol. 11, pp. 651-657, May 2001.

[7] C. Diou, L. Torres, and M. Robert, "A wavelet core for video processing," presented at the IEEE Int. Conf. Image Process., Sept. 2000.
[8] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels, and I. Bolsens, "Optimal memory organi -zation for scalable texture codecs in MPEG-4, IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 218-243, Mar. 1999.
[9] M. Vishwanath, R. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," IEEE Trans. Circuits Syst. II, vol. 42, pp. 305-316, May 1995.
[10] J. S. Fridman and E. S. Manolakos, "Discrete wavelet transform:Data dependence analysis and synthesis of distributed memory and control array architectures," IEEE Trans. Signal Proce -ssing, vol. 45, pp. 1291-1308, May 1997.
[11] T. Acharya, "A high speed systolic architecture for discrete wavelet transforms," in Proc. IEEE Global Telecommun. Conf., vol. 2, 1997, pp. 669-673.
[12] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," IEEE Trans. VLSI Syst., vol. 1, pp. 191-202, June 1993.
[13] A. Grzeszczak, M. K. Mandal, S. Panchanathan, and T. Yeap, "VLSI implementation of discrete wavelet transform," IEEE Trans. VLSI Syst., vol. 4, pp. 421-433, June 1996.
[14] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels, and I. Bolsens, "Optimal memory organi -zation for scalable texture codecs in MPEG-4," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 218-243, Mar. 1999.
[15] M. Ferretti and D. Rizzo, "A parallel archi -tecture for the 2-D discrete wavelet transform with integer lifting scheme," J. VLSI Signal Pro -cessing, vol. 28, pp. 165-185, July 2001.
[16] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform", IEEE Trans. on Sig -nal Processing, vol. 50, no. 4, April 2002.
[17] G. Dillen, B. Georis, J. D. Legat, and O. Canti -neau, "Combined Line-Based Architecture for the 5-3 and 9-7 Wavelet Tansform of JPEG 2000", IEEE Transactions on Circuit Syst. Video Technol., vol. 13, no. 9, Sep. 2003.
[18] http://www.barco.com/subcontracting/Downloads/ IPProducts/BA113FDWTFactS- heet.pdf
[19] http://www.cast-inc.com/cores/rc_2ddwt/ rc_2ddwt-a.pdf
[20] http://www.cast-inc.com/cores/lb_2dfdwt/ lb_2dfdwt-x.pdf
[21] http://www.cast-inc.com/cores/bb_2dfdwt/ cast_bb_2dfdwt-x.pdf
[22] http://www.amphion.com/cs6210.html
[23] I. Daubechies and W. Sweldens, "Factoring wa -let transforms into lifting schemes," J. Fourier Anal. Appl., vol. 4, pp. 247-269, 1998.
[24] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet construc -ons," in Proc. SPIE, vol. 2569, 1995, pp. 68-79.

저 자 소 개

서 영 호(종신회원)
1999년 2월 광운대학교 전자재료 공학과 졸업(공학사).
2001년 2월 광운대학교 대학원졸업(공학석사).
2000년 3월~2001년 12월 인티스닷컴(주) 연구원.
2003년 6월~2004년 6월 한국전기연구원 연구원.
2001년 3월~현재 광운대학교 전자재료공학과 박사과정.
<주관심분야: Image Processing/Compression, 워터마킹, 암호학, FPGA/ASIC 설계>

김 동 욱(종신회원)
1983년 2월 한양대학교 전자공학과 졸업(공학사).
1985년 2월 한양대학교 대학원 졸업(공학석사).
1991년 9월 Georgia공과대학 전기 공학과 졸업(공학박사).
1992년 3월~현재 광운대학교 전자재료공학과 정교수.
광운대학교 신기술 연구소 연구원.
2000년 3월~2001년 12월 인티스닷컴(주) 연구원.
<주관심분야: 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication>