Motion JPEG2000을 위한 리프팅 프로세서의 ASIC 설계

정회원 서 영 호*, 종신회원 김 동 욱*

ASIC Design of Lifting Processor for Motion JPEG2000

Young-Ho Seo*, Dong-Wook Kim** *Regular Members*

요 약

본 논문에서는 JPEG2000을 위한 새로운 리프팅 구조를 제안하고 ASIC으로 구현하였다. 동일한 구조의 반복적인 연산을 통해서 수행되는 리프팅의 특성을 이용하여 단위 연산을 수행할 수 있는 셀을 제안하고 이를 확장하여 전체 리프팅을 재구성하였다. 먼저, 리프팅 연산의 동작 순서를 분석하고 하드웨어의 구현을 고려한 인과성을 부여한 후 단위 셀을 최적화하였다. 제안한 셀의 단순한 확장을 통해서 리프팅 커널을 구성하고, 이를 이용하여 Motion JPEG2000을 위한 리프팅 프로세서를 구현하였다. 구현한 리프팅 커널은 최대 1024×1024 크기의 타일(Tile)을 수용할 수 있고, (9,7)필터를 이용한 손실압축과 (5,3)필터를 이용한 무손실압축을 모두 지원한다. 또한 입력 데이터율과 동일한 출력율을 가지고, 일정 대기지연 시간이후 4가지 부대역(LL, LH, HL, HH)의 웨이블릿 계수들을 연속적으로 동시에 출력할 수 있다. 구현한 리프팅 프로세서는 SAMSUNG의 0.35μm CMOS 라이브러리를 이용하여 ASIC 과정을 거쳤다. 약 9만개의 게이트를 사용하고, 곱셈기로 사용된 매크로 셀에 따라 차이는 있지만 약 150MHz 이상의 속도에서 안정적으로 동작이 가능하였다. 최종적으로 기존의 연구 및 상용 IP와의 비교에서도 종합적으로 우수한 성능을 보이는 것을 확인할 수 있었다.

Key Words : Lifting, Factorization, DWT, Line-based, Hardware, Filter.

ABSTRACT

In this paper, we proposed a new lifting architecture for JPEG2000 and implemented to ASIC. We proposed a new cell to execute unit calculation of lifting using the property of lifting which is the repetitious arithmetic with same structure, and then recomposed the whole lifting by expanding it. After the operational sequence of lifting arithmetic was analyzed in detail and the causality was imposed for implementation to hardware, the unit cell was optimized. A new lifting kernel was organized by expanding simply the unit cell, and a lifting processor was implemented for Motion JPEG2000 using it. The implemented lifting kernel can accommodate the tile size of 1024×1024, and support both lossy compression using the (9,7) filter and lossless compression using (5,3) filter. Also, it has the same output rate as input rate, and can continuously output the wavelet coefficients of 4 types(LL, LH, HL, HH) at the same time. The implemented lifting processor completed a course of ASIC using 0.35μm CMOS library of SAMSUNG. It occupied about 90,000 gates, and stably operated in about 150MHz though difference from the used macro cell for the multiplier. Finally, the improved operated in about 150MHz though difference from the used macro cell for the multiplier. Finally, the performance can be identified in comparison with the previous researches and commercial IPs.

*유한대학 전자정보과(yhseo@yuhan.ac.kr), **광운대학교 전자재료공학과({ddntlab, dwkim}@kw.ac.kr)
논문번호 : KICS2004-08-154, 접수일자 : 2004년 8월 14일
※본 연구보고서는 2004년도 중소기업청 산학연 공동기술개발 컨소시엄사업에 의해 지원되었음.

Ⅰ. 서 론

지난 10여년 동안 정보 및 데이터의 저장매체나 전송기술은 큰 발전을 이루었다. 또한 유선 통신에서 나아가서 무선 기반의 통신을 위한 제반적 기술 및 기반 인프라에 대한 구축이 괄목할 만한 성장을 이루었다. 이와 함께 사용자들은 영상이나 비디오 서비스와 같은 대용량의 정보와 우수한 서비스를 요구하고 있어 이러한 대용량의 데이터들의 효율적인 처리기술의 중요성이 높아지게 되었고 이에 대한 연구가 활발히 진행되어 왔다. 영상 정보를 효율적으로 처리하고자 하는 가장 대표적인 기술이 JPEG과 MPEG 및 H.26X의 표준들이며, 이들 표준들을 응용한 소프트웨어(software, S/W) 혹은 하드웨어(hardware, H./W) 제품들이 경쟁적으로 쏟아져 나오고 있다. JPEG 또는 MPEG은 이산 코사인 변환(Discrete Cosine Transform, DCT)을 기반으로 하는 기술로서 여러 각도의 기술적 진보에도 불구하고 블록효과라는 필연적인 단점을 갖고 있다. 이를 보완 및 대체하기 위한 기술이 최근 10여년간 연구되고 있는데 대표적인 것이 웨이블릿(wavelet)을 기반으로 하는 영상처리이다. DCT와는 다르게 이산 웨이블릿 변환(Discrete Wavelet Transform, DWT)은 블록효과를 제거할 수 있을 뿐 아니라 전체영상을 대상으로 인간의 시각에 따른 처리가 가능하여 JPEG2000[1]의 표준 변환으로 이미 지정되었다.

JPEG2000[1]의 저변의 확대와 함께 2D DWT에 대한 연구도 활발히 진행되고 있다. 최근에는 컨벌루션(convolution) 방식에 비해서 우수한 성능을 보이는 리프팅(lifting) 방식의 DWT 변환이 주로 연구되고 있다. 리프팅 기법은 기본적인 웨이블릿 변환(컨벌루션 방식)을 이용한 필터링 기법에 비해서 메모리량과 메모리에 대한 참조 횟수가 적고 정변환과 역변환이 동일한 구조로 이루어진다는 장점을 가지고 있다[2][3][4]. 대표적으로 연산의 효율성을 위해 입력 영상을 블록으로 구분지어 처리하는 방식[2]과 (5,3)필터를 이용하여 데이터를 인터리빙하는 구조[3]들이 연구되었다. 또한 EZW(Embedded Zero-tree Wavelet) 알고리즘 기반의 양자화 방식에 적합한 리프팅 구조[4]와 예측(predict)과 갱신(update)의 과정을 처리하는 단위 동작을 전체로 확장하는 구조[5]들이 제시되었다. JPEG2000에서 규정하고 있는 여러 종류의 필터에 대해서 적용 가능하도록 확장성을 가진 연산단위와 이들에 의한 리프팅 구조도 제안되었다[6]. 데이터패스부가 완벽히 파이프라인화 되어 있지 않으면 임계경로가 큰 단점을 가지고, 리프팅 연산을 그대로 H/W화하여 구조적으로 간결성을 유지하려면 연속된 입력 데이터에 대한 처리결과가 모호하게 되고 리프팅 연산과 내부 메모리와의 관계를 명확히 보이지 못한다[5][6]. 이를 해결하고자 전체적으로 파이프라인된 리프팅 연산과 확장성을 가진 H/W 구조[7]가 소개되었는데, 고속의 성능을 갖지만 리프팅 알고리즘을 그대로 사상하였기 때문에 파이프라인을 위해 많은 수의 레지스터가 요구되고 라인 기반의 리프팅 방식을 명확히 나타내지 못하는 단점이 있다. JPEG2000이 JPEG을 대체할 것으로 예상되고 있어 다양한 상용 IP(Intellectual Property)들이 개발되고 있다[8][9][10][11][12].

본 논문에서는 새로운 리프팅 구조를 제안하고 이를 ASIC으로 구현한다. 셀 기반의 리프팅 연산 방식을 재구성하여 확장이 가능한 구조를 제안하고, Motion JPEG2000을 위한 H/W에 구현에 이용이 가능하도록 IP화한 후 리프팅 프로세서로 구현한다. JPEG2000의 손실압축과 무손실 압축과정을 모두 수용할 수 있도록 (5,3)과 (9,7) 필터에 대한 동작을 모두 포함하는 구조를 갖도록 하는데, 이 경우 연산량의 차이에 따라서 (5,3) 필터의 경우 (9,7) 필터의 두 배에 해당하는 데이터율을 가질 수 있다. 이전 연구들에서 H/W기반의 리프팅 구조를 제시하고 있으나 동작의 복잡성이나 여러 부가회로의 부제 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 높은 효율을 가지면서 H/W의 구조와 제어를 단순화시키도록 한다.

본 논문은 다음과 같이 구성된다. 먼저 2장에서는 본 논문에서 사용하고 있는 리프팅 연산을 설명하고, H/W 구현을 위한 고정소수점의 정밀도를 모의실험을 통해서 분석한다. 다음으로 3장에서는 셀기반의 리프팅 구조를 제안한 후 프로세서를 구성하고, 4장에서는 앞장에서 제안한 구조에 대한 구현결과를 보이고, 이전연구 및 상용 IP들과 구조를 비롯한 성능을 비교한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 리프팅 알고리즘

본 장에서는 기본적인 리프팅 알고리즘에 대해서 설명하고, H/W를 통해 리프팅을 수행할 경우에 영

상의 열화 및 성능을 예측하고자 고정소수점 기반의 정밀도를 분석한다.

2.1 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[13][14]. 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 정의한다.

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

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

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

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

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

여기서 K는 상수이다. 그림 1은 식 (3)의 $\tilde{P_1}(z)$에 해당하는 일반적인 리프팅 과정을 나타내고 있는데, 리프팅을 이용한 웨이블릿 변환은 분할(split) 및 결합(merge), 예측(predict), 갱신(update), 조정(scaling)의 네 단계로 구성된다[13][14].

2.2 수 체계의 정밀도 분석

본 절에서는 리프팅 연산결과에 영향을 미치는 필터 계수와 웨이블릿 계수의 고정소수점 시뮬레이션을 통해서 H/W 구현에 따른 비트 제약이 성능에 미치는 영향을 분석하였고, 이를 바탕으로 H/W 구현을 위한 수 체계를 확립하였다.

수 체계의 정밀도 분석을 위한 고정소수점 곱셈 연산을 그림 2에 보이고 있는데, 17비트의 필터계수(정수 2비트, 소수 15비트)와 17비트의 웨이블릿 계수(정수 10비트, 소수 7비트)에 대한 연산이다. 곱

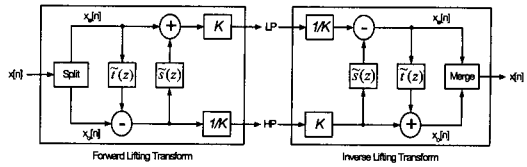


그림 1. 팩토링을 이용한 정방향/역방향 리프팅 방식의 구조도
Fig. 1. Forward/inverse lifting scheme using factoring

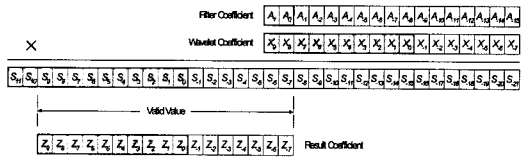


그림 2. 필터 연산을 위한 고정소수점 곱셈연산
Fig 2. Fixed-point multiplication for filter operation

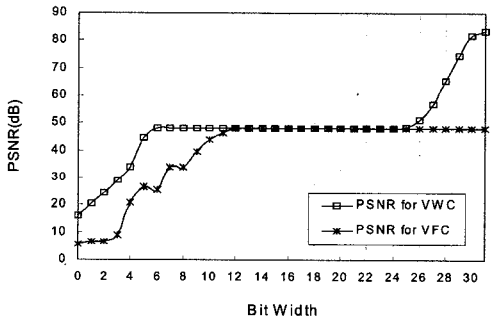


그림 3. 필터계수의 정밀도 분석
Fig. 3. Precision analysis of filter coefficient

셈 연산과정 중에 확장된 비트크기는 최종결과를 출력하기 전에 다시 정해진 크기로 정규화되어 일정 비트(Z9~Z-7)를 유지한다.

분석 결과를 그림 3에 나타냈는데, 웨이블릿 계수를 가변(VWC, Variable Wavelet Coefficient)으로 한 PSNR 결과와 필터 계수를 가변(VFC, Variable Filter Coeffieicnt)으로 한 PSNR 결과에 해당한다. 무손실 압축을 위해 사용되는 (5,3)필터는 실험에서 제외되었고 손실압축을 위한 (9,7)필터만이 사용하였다. 그림 3에 보이는 것과 같이 필터계수의 경우 소수점 이하 비트가 12비트 이상이면 약 50dB로 손실압축에 아무런 문제가 없을 정도의 성능을 보이고, 웨이블릿 계수의 경우에 소수점 이하 6비트 이상으로 표현된다면 50dB 이상의 좋은 성능을 나타낸다는 것을 알 수 있다. 그리고 그 이상의 정밀도를 부가한다고 해도 결과에는 거의 영향을 주지 않는다.

Ⅲ. 리프팅 커널의 제안

3.1 리프팅 연산의 재구성을 통한 LBFC의 구조

3.1.1 리프팅 연산의 재구성

리프팅 연산이 수행되는 절차를 도식적으로 나타내면 그림 4와 같다. (9,7) 필터를 이용한 리프팅은 리프팅 계수만 다르고 방식은 동일한 총 4단계의 곱셈과 덧셈과정을 거치는데, 이를 식 (5)~(8)에 나타냈다. 식 (8)의 결과는 식 (9)의 스케일링 과정을 거친다.

$$d_i^{(1)} = d_i^{(0)} + \alpha\,(s_i^{(0)} + s_{i+1}^{(0)}) \tag{5}$$

$$s_i^{(1)} = s_i^{(0)} + \beta\,(d_i^{(1)} + d_{i-1}^{(1)}) \tag{6}$$

$$d_i^{(2)} = d_i^{(1)} + \gamma\,(s_i^{(1)} + s_{i+1}^{(1)}) \tag{7}$$

$$s_i^{(2)} = s_i^{(1)} + \beta\,(d_i^{(2)} + d_{i-1}^{(2)}) \tag{8}$$

$$s_i = \zeta s_i^{(2)},\ d_i = d_i^{(2)}/\zeta \tag{9}$$

여기서 $s_i^{(0)}$과 $d_i^{(0)}$는 각각 x_{2i}과 x_{2i+1}에 해당한다. 또한 리프팅을 위한 계수값들은 표 1과 같이 정의되고, H/W에서는 마지막 열의 이진값을 사용한다.

리프팅은 동일한 연산구조를 가지므로 그림 4의 오른쪽 아래와 같은 하나의 단계로 대치시킬 수 있다. 마지막으로 하나의 단계로 사상된 리프팅 연산은 동일한 구조를 가지고 동일한 연산을 수행하면서 시간적으로 중복되지 않으므로 그림 4의 완편 아래쪽과 같은 하나의 단위 연산으로 사상될 수 있다. 이 경우 연속적으로 입력되는 데이터를 처리할 수 있어야 하는 기본사항을 만족시켜야 하는데, 그림 4에서 단위 연산을 시간에 따라 겹치지 않으면서 입력되는 데이터에 대해 리프팅 연산을 수행할 수 있도록 재구성하는 과정을 그림 5에 나타냈다.

단위 연산은 같은 시간에 곱셈과 덧셈이 동시에 수행되지 않으므로 하나의 곱셈기와 하나의 덧셈기로 모든 연산을 수행할 수 있는데, 그림 5의 (a)에 시간적인 클록순서에 따른 연산방식과 요구되는 레지스터 및 레지스터에 저장되는 데이터를 나타냈다. 그림 5에서 ⊗와 ⊕기호는 각각 곱셈과 덧셈 연산을 나타낸다. 그림 5의 (b)는 하나의 곱셈기와 두개의 덧셈기를 사용하여 리프팅 단위 연산을 수행하는 방법을 나타냈는데 (a)보다 규칙적으로 연산되는 것을 볼 수 있다. 그림 5의 리프팅의 재구성된 결과를 그림 6에 유사코드 형태로 나타내었다. LBFC_hor_type1()함수가 덧셈기를 한 개 사용한 경우이고, LBFC_hor_type2()함수가 두 개를 사용한 경우이다.

표 1. 리프팅 계수와 이진표현
Table 1. Lifting coefficient and binary form

Step	Coefficient	Decimal	Binary (17-bit fixed-point)
1	α	-1.58613432	10011010011111010
2	β	-0.052980118	11111100100111000
3	γ	0.8829110762	00111000100000011
4	δ	0.4435068522	00011100011000100
5	ζ	1.230174104	01001110101110110

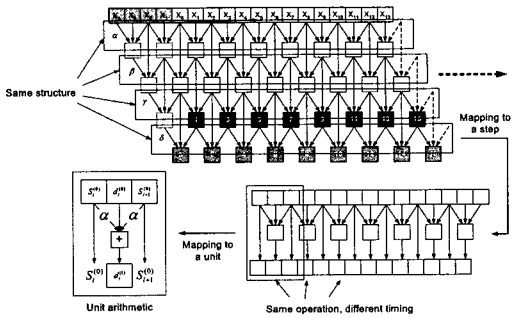
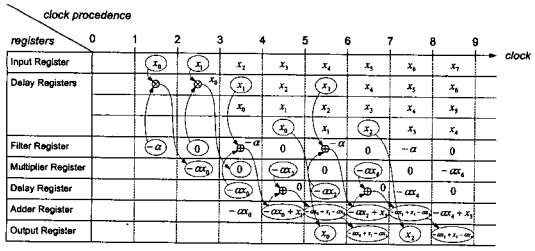


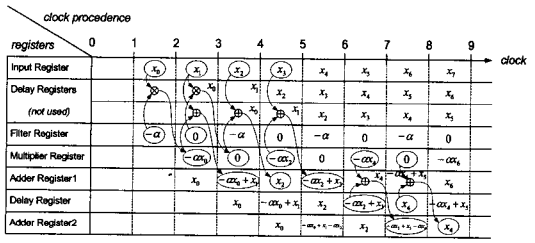
그림 4. 1차원 리프팅의 구조사상을 통한 단위연산
Fig. 4. Unit arithmetic of 1D lifting by mapping of structure

3.1.2 수평방향 LBFC의 구조

(5,3) 필터와 (9,7) 필터를 모두 수용하면서 연속적으로 입력되는 데이터를 처리할 수 있는 리프팅 단위 셀의 구조를 제안하고 이를 그림 7에 나타냈다. 그림 7의 (a)와 (b)는 그림 6의 재구성된 리프팅 연산코드를 H/W 구조로 변환한 것이다. 제안된 H/W를 리프팅 기반의 필터링 셀(LBFC, Lifting-Based Filtering Cell)이라고 한다. 그림 7의 (a)가 그림 6의 LBFC_hor_type1()함수를 H/W로 변환한 것이고 (b)가 LBFC_hor_type2()함수를 변환한 것이다. 두개의 H/W는 동일하게 5개의 파이프라인 단계를 가지지만, 그림 7의 (a)의 경우에 한 개의 덧셈기만을 쓰는 반면에 덧셈기로 입력되는 데이터를 시간적으로 배분하고자 세 개의 MUX와 3개의 레지스터가 추가된다. 표 2에 각각의 LBFC들이 사용하는 하드웨어 자원에 대해 요약하였다. SAMSUNG의 0.35μm CMOS 라이브러리를 사용할 경우에 표 2의 인스턴스들에 대한 게이트수를 표 3에 나타냈다. 세 개의 MUX와 레지스터의 게이트수를 합하면 420개인 반면에 한 개의 덧셈기는 105개이다. 또한 그림 7의 (a)와 (b)를 비교하면 (a)에 비해서 (b)가 더욱 간결한 구조를 가지고, MUX에 의한 데이터



(a)



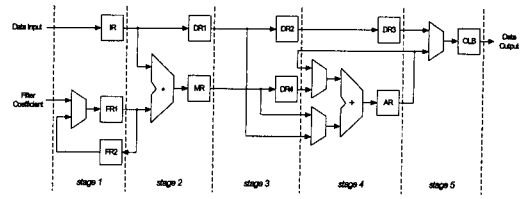
(b)

그림 5. 리프팅 연산의 시간 재구성 (a)덧셈기 한 개 사용 (b) 덧셈기 두 개 사용
Fig 5. Time rescheduling of lifting (a) use of one adder (b) use of two adders

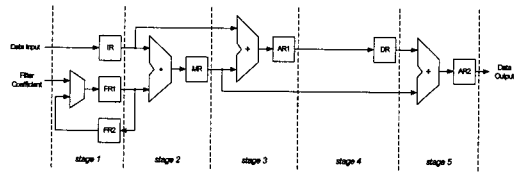
```
Define : IR        : Input Register
         MR        : Multiplier Register
         ARX       : Adder RegisterX
         DR        : Delay Register
         FR        : Filter Register
         OR        : Output Register
LBFC_hor_type1(){
   FR  <= {-a, 0};
   for(every clock) {
      IR <= Input Data;
      DR3 <= DR2 <= DR1 <= IR;
      MR <= IR*FR;
      DR4 <= MR;
      if (even) AR <= MR+AR;
                OR <= AR;
      else      AR <= DR1+DR4;
                OR <= DR3;
   }
}
LBFC_hor_type2() {
   FR  <= {-a, 0};
   for(every clock) {
      IR          <= Input Data;
      MR <= IR*FR;
      AR1 <= IR+MR;
      DR <= MR;
      AR2(OR) <= MR+DR;
      FR <= FR(Round Shift);
   }
}
```

그림 6. 재구성된 리프팅 연산 알고리즘
Fig. 6. Rearranged lifting arithmetic algorithm

제어가 필요하지 않아서 동작이 용이하다. 따라서 본 논문에서는 그림 7의 (b)에 해당하는 LBFC를 이용하여 ASIC으로 구현하였다.



(a)



(b)

그림 7. 수평방향 LBFC의 구조 (a) 방식1 (b) 방식2
Fig. 7. Architecture of horizontal LBFC(LBFC_hor) (a) type1 (b) type2

표 2. LBFC의 종류에 따른 하드웨어 자원
Table 2. Hardware resource for type of LBFC

Instance Name	Multiplier	Adder	Register	MUX
LBFC_hor_type1	1	1	10	4
LBFC_hot_type2	1	2	7	1

표 3. 기본 인스턴스의 게이트수
Table 3. Gate number of basic instance

Instance Name	Bit Width	Gate
Adder	17	105
Register	17	98
MUX	17	42

3.1.3 LBFC의 동작

그림 7의 LBFC에서 각 레지스터들은 데이터의 입력과 연산된 데이터의 저장 및 지연을 위한 버퍼링을 담당한다. 그림 7의 FR은 필터 계수를 저장하는데, 필터링 동작의 초기에 해당 필터 계수로 프로그래밍되고 시간순서에 따라서 필터계수와 영값을 번갈아 출력한다. 클록순서에 따라 레지스터들의 내용이 어떻게 바뀌는지를 표 4에 나타냈는데, 파이프라인 단계에 따라 5 클록 이후부터 결과가 출력되는 것을 확인할 수 있고 곱셈기의 지연만큼 임계경로를 가진다.

리프팅의 특성상 하나의 중간 연산결과가 여러 최종 연산결과에 영향을 미친다. 예를 들어 그림 4 혹은 표 4에서 $s_{i+1}^{(0)}$과 α의 곱셈결과는 $d_i^{(0)}$ 뿐만 아니라 $d_{i+1}^{(0)}$를 위한 연산에도 사용된다. 표 4에서 보이듯이 제안한 LBFC에서 다중으로 사용되는 중간 연산결과들을 구조화된 데이터 경로와 연산순서

표 4. LBFC의 동작
Table 4. Operation of LBFC

Clk	Input	IR	MR	AR1	DR	AR2
1	$s_{i+2}^{(0)}$	0	0	0	0	0
2	$d_{i+1}^{(0)}$	$s_{i+2}^{(0)}$	0	0	0	0
3	$s_{i+1}^{(0)}$	$d_{i+1}^{(0)}$	$\alpha s_{i+2}^{(0)}$	$s_{i+2}^{(0)}$	0	0
4	$d_i^{(0)}$	$s_{i+1}^{(0)}$	0	$d_{i+1}^{(0)}+\alpha s_{i+2}^{(0)}$	$s_{i+2}^{(0)}$	0
5	$s_i^{(0)}$	$d_i^{(0)}$	$\alpha s_{i+1}^{(0)}$	$s_{i+1}^{(0)}$	$d_{i+1}^{(0)}+\alpha s_{i+2}^{(0)}$	$s_{i+3}^{(0)}$
6	$d_i^{(0)}$	$s_i^{(0)}$	0	$d_i^{(0)}+\alpha s_{i+1}^{(0)}$	$s_{i+1}^{(0)}$	$\alpha s_{i+1}^{(0)}+d_{i+1}^{(0)}+\alpha s_{i+2}^{(0)}$
7	$s_{i+1}^{(0)}$	$d_i^{(0)}$	$\alpha s_i^{(0)}$	$s_i^{(0)}$	$d_i^{(0)}+\alpha s_{i+1}^{(0)}$	$s_{i+1}^{(0)}$
8	$d_{i+1}^{(0)}$	$s_{i+1}^{(0)}$	0	$d_i^{(0)}+\alpha s_i^{(0)}$	$s_i^{(0)}$	$\alpha s_i^{(0)}+d_i^{(0)}+\alpha s_{i+1}^{(0)}$
9	$s_{i+2}^{(0)}$	$d_{i+1}^{(0)}$	$\alpha s_{i+1}^{(0)}$	$s_{i+1}^{(0)}$	$d_i^{(0)}+\alpha s_i^{(0)}$	$s_i^{(0)}$
10	$d_{i+2}^{(0)}$	$s_{i+2}^{(0)}$	0	$d_{i+1}^{(0)}+\alpha s_{i+1}^{(0)}$	$s_{i+1}^{(0)}$	$\alpha s_{i+1}^{(0)}+d_i^{(0)}+\alpha s_i^{(0)}$
11	$s_{i+3}^{(0)}$	$d_{i+2}^{(0)}$	$\alpha s_{i+2}^{(0)}$	$s_{i+2}^{(0)}$	$d_{i+1}^{(0)}+\alpha s_{i+1}^{(0)}$	$s_{i+1}^{(0)}$
12	$d_{i+3}^{(0)}$	$s_{i+3}^{(0)}$	0	$d_{i+2}^{(0)}+\alpha s_{i+2}^{(0)}$	$s_{i+2}^{(0)}$	$\alpha s_{i+2}^{(0)}+d_{i+1}^{(0)}+\alpha s_{i+1}^{(0)}$

에 따라 최소의 대기지연시간만을 가지면서 여러 연산에 관여한다. 따라서 동일한 결과를 가지는 중간 연산들은 두 번 이상 수행되지 않아서 최소의 연산수와 최적의 생존시간을 가지고 효율적인 리프팅을 구성한다.

3.2 라인기반 수직 리프팅을 위한 LBFC 구조

3.2.1 수직방향 리프팅의 경계처리

JPEG2000에서는 영상의 경계에서 대칭적인 확장 방식을 채택하고 있다. 수평방향으로 리프팅을 수행할 경우에는 단지 4개의 부가적인 화소와 그에 따르는 중간 결과들만 저장하고 있으면 되지만 수직 방향의 리프팅을 수행할 경우에는 입력영상 한줄 너비의 4배에 해당하는 부가적인 화소와 그에 따르는 수많은 중간 연산결과들을 저장해야한다. 이러한 부가적인 H/W의 추가를 막기 위해서는 리프팅 변환 레벨에 따라 메모리로부터 불규칙적으로 데이터를 호출해야하고, 불규칙적으로 주소를 호출하기 위해서 각 주소를 저장해야한다. 따라서 주소를 저장하기 위한 저장소가 증가하고 제어가 복잡해진다. 따라서 제어의 복잡도를 최소화하면서 대칭적인 확장방식과 유사한 결과를 보이는 방법을 제안하고, 이를 그림 8에 도식적으로 나타냈다.

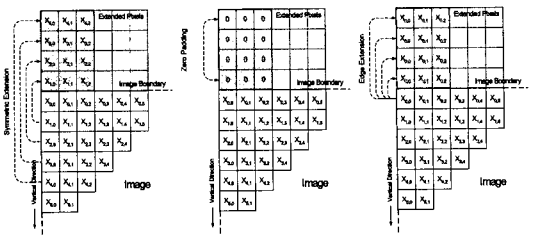


그림 8. 제안한 경계처리 방식 (a) 대칭적 확장 (b) 영값 확장 (c) 가장자리 확장
Fig. 8. Proposed boundary processing (a) symmetric extension (b) zero padding (c) edge extension

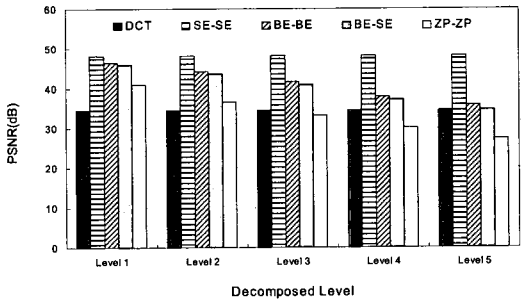
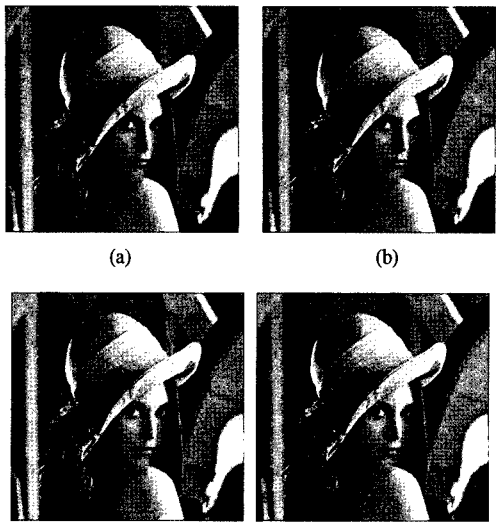


그림 9. 고정소수점 검증을 통한 경계처리 방식들의 PSNR 결과
Fig. 9. PSNR result of boundary processing methods by fixed-point simulation

그림 8의 (a)는 JPEG2000 표준에서 채택하고 있는 대칭적인 확장방법(SE, Symmetric Extension)이고, (b)는 경계를 가장 쉽게 확장할 수 있는 방법인 영값 확장방법(ZP, Zero Padding)이다. 그림 8의 (c)가 본 논문에서 제안하고자 하는 가장자리 확장 방법(EE, Edge Extension)으로 영상의 경계에 있는 화소들을 연속적으로 확장하는 방식이다. 경계화소 확장 방식들간의 성능을 그림 9에 나타냈다. 그림 9는 아래와 같이 5가지 방식에 대한 결과를 나타낸 것으로 그래프의 각 항목은 (인코딩-디코딩)에 대한 PSNR 결과이다. 512×512 크기의 500개 영상을 대상으로 고정소수점 시뮬레이션을 수행한 것이다. 5레벨까지 리프팅을 수행했을 경우에 ZP로 인코딩과 디코딩을 수행한 결과를 제외하고는 DCT보다 높은 결과를 보였고, 그림 10에 보인것과 같이 실제 화질은 크게 차이가 없음을 알 수 있다. 따라서 제안한 방식은 SE 혹은 EE 방식으로 디코딩을 하더라도 수용할만한 성능을 보이므로 JPEG2000 압축을 위한 좋은 방법이 될 수 있다.



(a) (b) (c) (d)

그림 10. 제안한 경계처리에 의한 영상결과 (a) 원영상 (b) SE-SE (48.1308dB) (c) EE-EE(38.9297dB) (d) EE-SE (39.3947dB)
Fig. 10. Image result by proposed boundary processing (a) original image (b) SE-SE (48.1308dB) (c) EE-EE (38.9297dB) (d) EE-SE(39.3947dB)

3.2.2 라인방식의 리프팅

라인방식의 웨이블릿 변환기법은 2차원 데이터를 대상으로 이루어지는데, 수평방향의 1차원 변환을 수행하면서 2차원 변환을 수행할 수 있는 결과가 마련되면 1차원 변환과 함께 수직방향의 2차원 변환을 동시에 수행하는 것이다. 이와 유사한 과정을 그림 11에 보이고 있는데, 수평방향의 리프팅이 수행되면 일정 대기지연시간 이후에 저주파와 고주파 계수에 해당하는 Lx,y와 Hx,y가 번갈아가며 매 클록마다 출력된다. 출력된 계수들은 버퍼링 과정없이 곧바로 수직방향의 리프팅 연산에 사용된다. 수평방향의 리프팅 결과가 다섯 번째 줄(L4,0)에 해당하면 일정 대기지연시간 후에 유효한 수직방향의 리프팅 결과(LL0,0, LH0,0)를 출력한다. 3.2.1에서 설명한 것과 같이 수직방향의 리프팅 연산을 수행할 때에 경계는 가장자리 화소를 사용한다.

3.2.3 수직 LBFC의 구조

본 논문에서는 수직방향의 리프팅을 수행하기 위해서 라인 기반의 필터링을 채택하고 있으므로 그림 7의 (b)에서 각 레지스터들이 그림 12와 같이 라인 버퍼(Line buffer)로 대체된다. 수평방향을 위한 리프팅과 동일한 구조를 가지고 있고, 하나의 블록으로 네 부대역의 계수들을 순서대로 출력(LLx,y→LHx,y→HLx,y→HHx,y)한다.

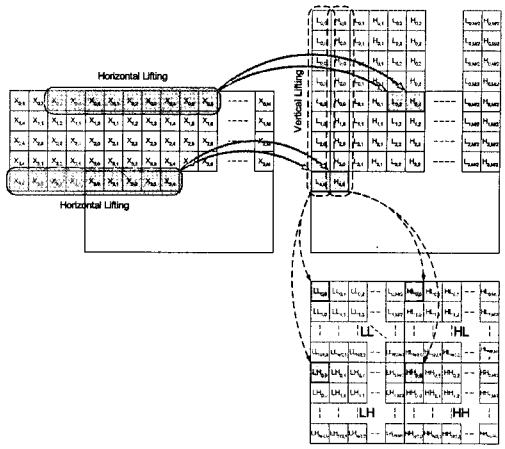


그림 11. 라인기반의 수직방향 리프팅 연산
Fig. 11. Line-based vertical lifting arithmetic

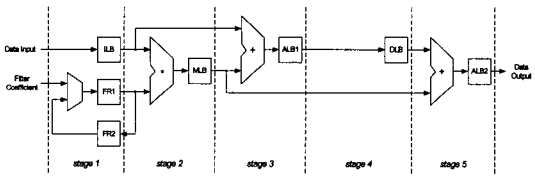


그림 12. 수직방향 LBFC의 구조
Fig. 12. Architecture of vertical LBFC(LBFC_ver)

3.3 경계처리 블록(BPU)의 구조

JPEG2000 표준안에서는 영상의 경계에서 대칭적으로 확장하는 방식을 채택하고 있는데, 이를 위한 H/W 구조(BPU, Boundary Processing Unit)를 그림 13에 나타냈다. LBFC와는 독립적으로 동작이 가능하고 입력되는 직렬 데이터에 대해서 간단한 버퍼링 동작을 수행하여 앞부분과 뒷부분에 4개의 데이터를 덧붙임으로써 대칭적으로 데이터를 확장한다. 그림 13에서 R4와 R5 레지스터를 기점으로 MUX를 통해서 서로 데이터를 대칭적으로 전달할 수 있는 것을 볼 수 있다.

3.4 리프팅 커널의 구조

그림 14는 그림 7과 12의 LBFC를 이용한 리프팅 방식의 필터링 구조인 LFDWT(Lifting Filter for DWT)와 리프팅 커널(Lifting Kernel)을 나타낸다. (9,7) 필터를 이용하는 경우 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

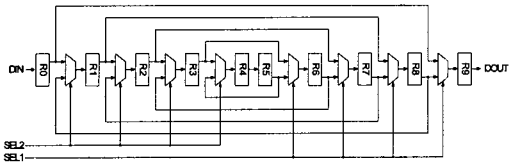


그림 13. BPU의 구조
Fig. 13. Architecture of BPU

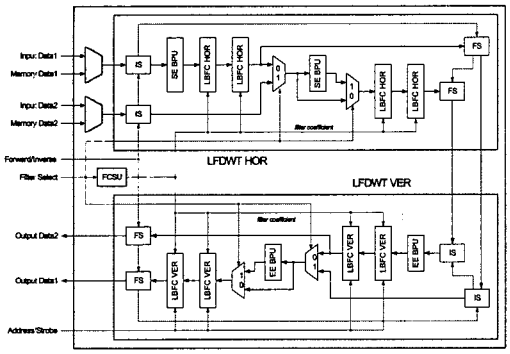


그림 14. LFDWT를 이용한 Lifting Kernel의 구조
Fig. 14. Architecture of Lifting Kernel using LFDWT

을 얻을 수 있는데, 그림 14에서 앞의 두 LBFC와 MUX 다음의 두 LBFC에 의해서 병렬적인 필터링을 수행하고, 이 때 MUX에 의해서 3번째 LBFC로 직접 입력한다. 리프팅을 이용한 필터링은 레벨 단위로 이루어지고 f의 속도로 입력된 "Din"은 연산을 거친 후 f의 속도로 네 개의 출력을 직렬로 출력하여 결국 입력과 동일한 f의 출력율을 가진다. IS와 FS는 표 1의 ζ값에 의한 역방향 및 순방향 스케일링 블록을 각각 나타낸다.

3.5 제안한 리프팅 프로세서

구현된 리프팅 프로세서의 구조를 그림 15에 나타냈다. A/D 변환기와의 인터페이스를 위한 A/D Interface, SDRAM의 제어를 위한 SDRAM Controller, SDRAM과의 데이터 입출력을 위한 SDRAM Buffer, 리프팅을 수행하는 Lifting Kernel, 명령어를 저장하는 Programming Register, 그리고 전체 제어를 위한 Main Controller로 구성된다. "Data Input"을 통해서 직접 데이터를 입력받거나 혹은 A/D 변환기로부터 데이터를 입력받을 수 있다. 또한 직접 데이터를 입력받지 않고 SDRAM Controller를 이용하여 SDRAM에 저장시킨 데이터를 처리할 수도 있는데, 이 경우 SDRAM으로부터 입력된 데이터의 처리와 함께 다음 프레임을 SDRAM에 저장하여 프레임 단위의 실시간 처리 동작을 수행할 수 있다. 전체적인 동작은 "I2C" 포트를 통해

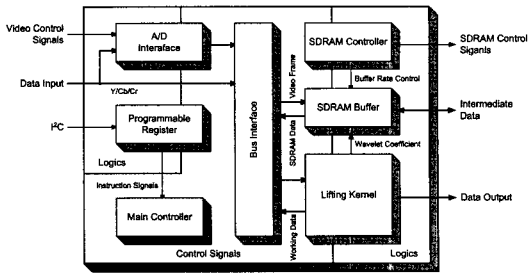


그림 15. 리프팅 프로세서의 구조
Fig. 15. Architecture of lifting processor

서 Programming Register를 어떻게 구성하느냐에 따라 결정되고, Main Controller가 전체적인 제어를 담당한다.

Ⅳ. 구현 및 결과

본 논문에서 제안한 리프팅 커널은 ASIC 및 FPGA 환경으로 모두 구현하였다. ASIC은 삼성의 0.35μm CMOS 라이브러리를 이용하여 구현하였고, FPGA는 Altera사의 APEX를 타겟으로 하였다. 리프팅 알고리즘은 C++언어를 이용하여 검증하였고 이를 Verilog-HDL을 사용하여 RTL 수준의 H/W로 사상하였다. 제안한 리프팅 커널은 JPEG2000 표준을 위한 H/W이므로 입력 영상을 타일 단위로 처리하고 라인버퍼는 최대 1024×1024 크기의 타일을 수용할 수 있다.

4.1 하드웨어 자원

합성 후에 측정된 H/W 자원을 표 5에 자세히 나타내었다. 전체 회로는 약 9만개의 게이트로 구성되는데, 이 중에서 Lifting Kernel이 약 8만 4천개의 게이트를 점유하고 있고 SDRAM Controller가 5천개의 게이트를 점유한다. 구현된 회로에서 Line Buffer는 메모리가 아닌 레지스터를 사용하였는데 이는 Line Buffer의 용량이 비교적 크지 않으면서 다수(총 4×4개)가 사용되기 때문이다. 이 경우에 메모리 컴파일러를 통해 제공되는 SRAM 혹은 DRAM을 사용하는 것이 불합리하다. 또한 Lifting Kernel에 사용되는 Line Buffer의 크기가 대용량을 요구하는 것이 아니므로 레지스터로 구현이 가능하고 라이브러리에 구애받지 않고 쉽게 이식하는데도 유리하다.

4.2 ASIC 및 FPGA 구현과 성능 비교

이미 설명한 바와 같이 제안된 리프팅 커널은

표 5. 설계된 Lifting Processor의 하드웨어 자원
Table 5. Hardware resource of lifting processor

#	Instance Name	Sub Instance		Gate Size
1	LFDWT_HOR_TOP			17,926
2		LFDWT_HOR		14,336
3		Scaling Block, Logics		3,590
4	LFDWT_VER_TOP			65,872
5		LFDWT_VER		62,118
6			Line Buffer	45,760
7			Arithmetic	16,358
8		Scaling Block, Logics		3,745
9	SDRAM Controller			5,439
10	Lifting Kernel			83,798
11	Lifting Processor			89,239

ASIC과 FPGA로 모두 구현하였다. 구체적인 결과와 다른 연구 및 상용 제품과의 비교를 표 6과 표 7에 각각 나타냈다. 첫 번째 열의 H/W를 제외하고는 모두 리프팅 알고리즘과 관련된 항목들이다. 일반적으로 알려진 것과 같이 컨벌루션 방식이 많은 H/W 자원을 사용한다. 뿐만 아니라 가시적으로 나타내지는 않았지만 실제 구현에서 가장 중요한 요소인 메모리 접근횟수도 리프팅 방식에 비해 컨벌루션이 상당히 많은 것이 일반적이다. H/W양은 두 번째 열의 H/W가 가장 작은데, 이는 (5,3) 필터만이 사용하기 때문이다. (9.7) 필터를 이용한 필터링이 (5.3) 필터를 이용한 필터링에 비해서 리프팅의 경우에 약 2배 이상의 복잡도를 가지는 것을 고려하여 다른 H/W와 동일한 기준을 적용하면 약 2배 정도로 H/W양이 증가해야 한다. 두 가지 필터를 수용하면서 H/W 양이 가장 작은 것이 세 번째 열의 H/W인데, 출력 데이터율이 다른 것들에 비해서 1/2인 단점이 있다. 이 구조는 데이터율을 높이기 위해서 수직 필터링 블록을 하나 증가시켜야 하는데, 이 경우에 곱셈기와 덧셈기가 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 첫 번째 열의 H/W와 제안한 구조가 비교적 단순한 제어 특성을 보이고 나머지 H/W 구조들은 상당히 복잡한 제어 관계를 갖는다. 표 6에서는 이전 연구결과들과 동일한 조건으로 비교하기 위해 연산을 수행하는 부분만 가장 간략화된 형태로 결과를 보이고 있다. 필터의 적응성, 처리할 수 있는 영상크기에 따른 저장 공간(메모리의 양), 동일 성능에 대한 곱셈기 및 덧셈기의 수, 그리고 제어의 복잡도 등을 비교할 때 구현된 리프팅 커널이 가장 우수한 것을 볼 수 있다.

JPEG2000의 저변이 확대되면서 상용화된 다양한 리프팅 커널이 출시되고 있는데, 표 7에서 이러한 몇 가지 IP들과 비교하였다. 표 7의 IP들은 JPEG 2000을 위한 것이므로 표 6의 연구들과 달리 입력 영상을 타일 단위로 규정하고 있고 제품으로 이미 판매하는 것들이다.

표 7에서 Tile size는 입력 영상크기를 뜻하고 이는 메모리의 크기와 직결된다. Transform은 순방향 및 역방향 변환을 수행할 수 있는지를 나타내는 것이고, Filter는 어떤 필터를 지원할 수 있는지를 나타낸다. 그리고 구현 환경과 게이트 수, 동작 속도 등을 비교하고 있다. 먼저 BB-2DFDWT의 경우 (5,3) 필터만을 지원하면서 순방향 필터링만 가능한 Soft IP 형태로서 성능이 가능 낮은 것을 볼 수 있다. 또한 RC_2DDWT 및 LB-2DFDWT도 BA113 FDWT와 CS6210에 비해서 성능이 낮은 분야에 사용되는 IP들이다. BA113FDWT와 CS6210 및 본 논문의 리프팅 커널을 비교하면 H/W 양에 비해서 순방향 필터링만 수행하는 BA113FDWT가 가장 많

표 6. 제안된 H/W의 자원 사용
Table 6. Resource usage of proposed H/W

Arch.	Filter	Image Size	*	+	Storage Size	Output Rate	Operation Frequency	Gate Count	Control Complexity
[1]	(9,7)	256×256	36	36	18.5Kbit	f	-	-	Simple
	(5,3)	256×256	36	36	17.5Kbit	f	-	-	Simple
[2]	(5,3)	129×129	4	8	236 Kbit	f	200Mhz ($0.18\mu m$)	15,000 ($0.18\mu m$)	Complex
[3]	(9,7)/ (5,3)	256×256	8	16	57.6Kbit	f/2	110Mhz (FPGA)	2,363 SL (+3,449Reg)	Complex
Ours	(9,7)/ (5,3)	1024×1024	12	24	256Kbit	f	>150Mhz ($0.35\mu m$)	32,262 ($0.35\mu m$)	Simple

표 7. 상용 IP들과 성능 비교
Table 7. Performance comparison of commercial IPs

Design	Tile Size	Tran-sform	Filter	Target Technology	Gate Count	Clock (MHz)	Memory
BA113FDWT[4]	128× 128	F	(9,7), (5,3)	Altera EP1S2	4,291 LE	130	2 MRAM
				Xilinx XC2V100	3,381 SL	135	42 RAMB16
				ASIC(0.18μm)	50,000	205	112Kbits+ 512Kb
RC_2DDWT[5]	512× 512	F/I	(9,7)	Altera	3,280 LE	65	External RAM
				Xilinx Vertex II	1,698 SL	95	External RAM
LB-2DFDWT[6]	256× 256	F	(9,7)	Xilinx Vertex II	2,227 SL	51	14 Block RAM
BB-2DFDWT[7]	-	F	(5,3)	Xilinx Vertex II	946 SL	52	10 Block RAM
CS6210[8]	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(0.18μm)	55,000	150	50KB
Ours	1024× 1024	F/I	(9,7), (5,3)	Altera APEX20K	5820 LE	52	128 ESB
				ASIC(0.35um)	38,038	150	256Kbit
					112,965	150	0Kbit

은 자원을 사용하고 있으며 리프팅 커널이 순방향 및 역방향 필터링을 모두 지원하면서 ASIC의 경우 38,038개(다른 회로와 마찬가지로 라인 버퍼를 메모리로 사용할 경우로 스케일링 블록 포함)로 가장 작고, FPGA의 경우 5,820개의 LE(Logic Element)로 두 번째로 작은 자원을 사용하고 있다. 동작 속도의 경우 BA113FDWT와 CS6210이 제안된 H/W에 비해 높은 것을 볼 수 있으나, 이는 공정 및 FPGA 디바이스의 선택에 따라 변경될 수 있다. 또한 메모리의 사용을 보면 동일한 타일 크기를 지원한다고 가정할 경우에 BA113FDWT는 112Kbit×4이고 RC_2DDWT은 50K×8×4의 메모리를 사용하므로, 256Kbit를 사용하는 제안한 리프팅 커널이 가장 작은 메모리를 필요로 한다.

4.3 P&R 결과

Apollo를 이용하여 PNR을 수행하였고 DRC 및 LVS 과정을 에러없이 결과를 추출하였다. 전체 chip 크기는 2655×2655이고 설계된 core의 utilization은 약 80%에 해당한다. 그림 16에 설계된 H/W의 P&R의 결과 레이아웃을 나타냈다.

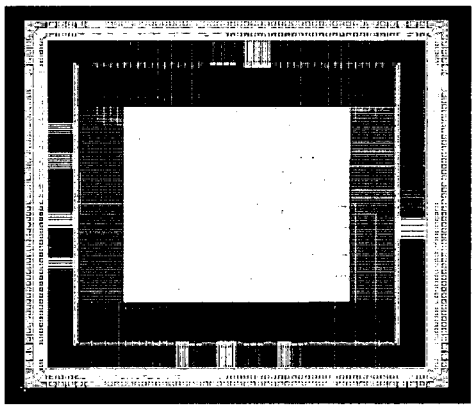


그림 16. P&R 후의 레이아웃 결과
Fig. 16. Layout result after P&R

V. 결 론

본 논문에서는 적은 H/W 자원과 효율적인 동작 성능을 보이는 새로운 형태의 리프팅 구조를 제안하고 ASIC으로 구현하였다. 리프팅 연산을 시간 순서에 따라서 분석한 후 H/W를 위한 구조로 재구성하였고, 재구성한 리프팅을 셀 기반의 확장이 용이한 H/W로 구현하였다. 구현한 리프팅 커널은 최대 1024×1024 크기의 타일을 수용할 수 있고, 입력 데이터율과 동일한 출력율로 일정 대기지연 시간이 후 4가지 부대역(LL, LH, HL, HH)의 웨이블릿 계수들을 순차적으로 출력할 수 있었다. 또한 JPEG 2000에서 요구하는 (9,7)필터와 (5,3)필터를 모두 사용할 수 있어 손실압축과 무손실압축을 모두 지원할 수 있었다. SAMSUNG의 0.35μm CMOS 라이브러리를 이용하여 ASIC으로 구현된 리프팅 프로세서는 약 9만개의 게이트를 점유하면서 150MHz 이상의 속도에서 안정적으로 동작하였다. 구현된

H/W는 기존의 연구와 IP와 비교해서 우수한 특성을 가지고 있으면서 Motion JPEG2000에서 요구하는 사항을 모두 충족시킬 수 있어 추후 다양한 영상압축 분야에 적용될 수 있을 것으로 사료된다.

참 고 문 헌

[1] M. Boliek, C. Christopoulos, and Eric Majani, "JPEG 2000 part-I final draft international standard", ISO/IEC JTC1/SC29 WG1, 24 Aug. 2000.

[2] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," IEEE Trans. VLSI Syst., vol. 1, pp. 191-202, June 1993.

[3] 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.

[4] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels, and I. Bolsens, ""Optimal memory organization for scalable texture codecs in MPEG-4," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 218-243, Mar. 1999.

[5] M. Ferretti and D. Rizzo, "A parallel architecture for the 2-D discrete wavelet transform with integer lifting scheme," J. VLSI Signal Processing, vol. 28, pp. 165-185, July 2001.

[6] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform", IEEE Trans. on Signal Processing, vol. 50, no. 4, April 2002.

[7] G. Dillen, B. Georis, J. D. Legat, and O. Cantineau, "Combined Line-Based Architecture for the 5-3 and 9-7 Wavelet Tansform of JPEG2000", IEEE Transactions on Circuit Syst. Video Technol., vol. 13, no. 9, Sep. 2003.

[8] http://www.barco.com/subcontracting/Downloads/IPProducts/BA113FDWTFactSheet.pdf

[9] http://www.cast-inc.com/cores/rc_2ddwt/rc_2dd-wt-a.pdf

[10] http://www.cast-inc.com/cores/lb_2dfdwt/lb_2d-fdwt-x.pdf

[11] http://www.cast-inc.com/cores/bb_2dfdwt/cast_bb_2dfdwt-x.pdf

[12] http://www.amphion.com/cs6210.html

[13] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting schemes," J. Fourier Anal. Appl., vol. 4, pp. 247-269, 1998.

[14] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in Proc. SPIE, vol. 2569, 1995, pp. 68-79.

서 영 호(Young-Ho Seo) 정회원

1999년 2월 광운대학교 전자재료공학과(공학사)
2001년 2월 광운대학교 대학원(공학석사)
2000년 3월~2001년 12월 인티스닷컴(주) 연구원
2003년 6월~2004년 6월 한국전기연구원 연구원
2004년 8월 광운대학교 대학원졸업(공학박사)
2004년 9월~2004년 11월 유한대학 겸임교수
2004년 12월~현재 유한대학 연구교수
<관심분야> Image Processing/Compression, 워터마킹, 암호학, FPGA/ASIC 설계

김 동 욱(Dong-Wook Kim) 종신회원

1983년 2월 한양대학교 전자공학과(공학사)
1985년 2월 한양대학교 대학원(공학석사)
1991년 9월 Georgia공과대학 전기공학과(공학박사)
1992년 3월~현재 광운대학교 전자재료공학과 정교수. 광운대학교 신기술 연구소 연구원.
2000년 3월~2001년 12월 인티스닷컴(주) 연구원.
<관심분야> 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication