JPEG2000 영상압축을 위한 리프팅 설계 알고리즘을 이용한 2차원 이산 웨이블릿 변환 프로세서의 FPGA 구현에 대한 연구

송 영 규, 고 광 철, 정 제 명
한양대학교 전자전기컴퓨터공학부
전화 : 02-2290-0348 / 핸드폰 : 019-214-0930

A study on a FPGA based implementation of the 2 dimensional discrete wavelet transform using a fast lifting scheme algorithm for the JPEG2000 image compression

Abstract

The Wavelet Transform has been applied in mathematics and computer sciences. Numerous studies have proven its advantages in image processing and data compression, and have made it a basic encoding technique in data compression standards like JPEG2000 and MPEG-4. Software implementations of the Discrete Wavelet Transform (DWT) appears to be the performance bottleneck in real-time systems in terms of performance. And hardware implementations are not flexible. Therefore, FPGA implementations of the DWT has been a topic of recent research. The goal of this thesis is to investigate of FPGA implementations of the DWT Processor for image compression applications. The DWT processor design is based on the Lifting Based Wavelet Transform Scheme, which is a fast implementation of the DWT. The design uses various techniques. The DWT Processor was simulated and implemented in a FLEX FPGA platform of Altera.

I. 서론

웨이블릿 변환은 높은 에너지 집중성과 다해상도와 같은 우수한 특성 때문에 영상 압축, 영상 화질 개선, 영상 추정, 특징 추출, 잡음 제거 등의 다양한 분야에서 최근 많이 도입되고 있다. 그 중에서 리프팅 이중 직교 웨이블릿 변환은 시간축 또는 공간축 상에서의 접근을 통해서 원하는 웨이블릿 변환을 구현하는 방법으로 리프팅 구조의 특성상 계산량이 필터뱅크에 의한 방법보다 절반 정도로 적고, 보조 메모리가 필요하지 않아서 메모리를 적게 사용한다. 그리고 무손실 영상 압축을 위한 정수 대 정수 변환이 용이하게 구현되며, 역 변환이 쉽게 구현되고 순방향 변환과 복잡성이 동일하다는 장점을 가지고 있다.

이산 웨이블릿 변환의 소프트웨어 구현은 유연성이 있어 필터에 따른 확장과 이용이 쉽지만 실시간 시스템에서는 병목현상 때문에 느려진다. 하드웨어 구현은 성능면에서 뛰어나지만 유연성이 없다. 이 두 가지의 타협점으로 FPGA로 구현하는 방법이 있다. 이 방법은 재구성이 가능한 하드웨어로 설계하는 것이기 때문에 유연성을 보장하면서 하드웨어 구현의 장점인 고성능도 보장할 수 있다. 그러나 FPGA로 구현하기 위해서는 복잡도가 구현이 가능할 수 있도록 간단해야 한다.

본 논문에서는 Daubechies 4 필터를 사용하여 이산 웨이블릿 변환을 효과적이고 빠르게 동작할 수 있도록 하는 구조를 제안한다. 그 방법으로 이산 웨이블릿 변환을 빠르게 동작 시킬 수 있는 방법 중에 하나인 Lifting Scheme을 사용한다. 그리고 변환을 빠르게 하기 위해 여러가지 설계방법을 사용한다. 이산 웨이블릿 변환 프로세서는 VHDL로 설계한 후 Altera FPGA에 합성해서 동작을 검증하였다.

II. 웨이블릿 변환

2.1 이산 웨이블릿 변환

웨이블릿은 신호 혹은 함수를 표현하거나 분석하는 것에 사용할 수 있는 수학 함수이다. 주파수 분해 만

을 가질 뿐 시간 분해는 가지지 않는 푸리에 변환의 단점들을 극복하기 위한 해법이기도 하다.

이산 웨이블릿 변환은 근사(approximation) 신호 L (저역통과 필터의 결과)과 세부(detail) 신호 H (고역통과 필터의 결과)를 구하기 위해서 주로 컨벌루션(convolution)과 필터 쌍의 다운샘플링(down sampling)을 통해 계산한다.

그러므로 모웨이블릿의 팽창(dilations)과 천이(translations)는 다음과 같이 정의 된다.

$$\Psi_{j,k}(t) = 2^{-\frac{j}{2}}\Psi(2^{-j}t - k) \qquad [1]$$

여기에서 j는 스케일링(scaling) 인수이고 k는 천이(translation) 인수이다.

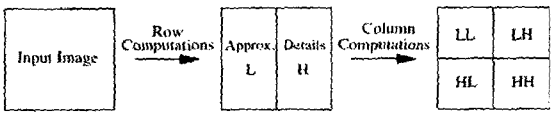
순방향 변환과 역방향 변환은 다음 식[2], [3]으로 계산된다.

$$C_{j,k} = \int_{-\infty}^{\infty} f(t)\Psi_{j,k}(t)dt \qquad [2]$$

$$f(t) = \sum_{j,k} C_{j,k}\Psi_{j,k}(t) \qquad [3]$$

이산 웨이블릿 변환은 여러가지 시간 해상도에 따라 다른 주파수 대역으로 신호를 분석한다. 이런 원리를 다해상도해석 (Multi-Resolution Analysis)이라고 하며 신호를 다양한 여러가지의 창 크기를 가지고 해석한다는 것을 의미한다. 다해상도 분해는 근사 신호를 컨벌루션과 다운샘플링을 해서 다시 근사 신호, 세부 신호로 반복함으로써 얻어진다.

이차원 신호의 경우는 그림 1과 같이 일차원 필터를 사용해서 먼저 수평방향으로 분해를 하고 다음에 수직방향으로 분해를 해서 계산할 수 있다.



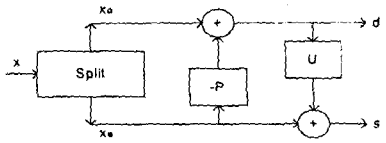
<그림 1. 이차원 이미지의 일단계 웨이블릿 분해>

다단계로의 분해는 저역통과 신호인 LL을 반복해서 분해하면 얻어진다.

2.2 the Lifting Scheme

컨벌루션을 이용한 웨이블릿 변환은 입력신호의 각 대역을 얻어 낼 수 있으나, 계산량이 많다는 단점이 있다.

보다 빠른 웨이블릿 변환을 하기 위한 방법으로 리프팅(lifting)을 이용한 방법이 있다. 기존의 필터 뱅크를 이용한 방법보다 일반적인 구조를 제공할 뿐만 아니라 구조의 특성상 필터 뱅크 접근 방법보다 계산량을 줄일 수 있고, 메모리를 적게 사용한다.



<그림 2. 리프팅기반 웨이블릿 변환>

그림 2에서 개략적으로 보여 주듯이 리프팅은 3단계로 구분할 수 있다. 먼저, 입력신호를 짝수번째와 홀수번째 신호로 나누는 분할 단계(splitting step), 고대역 신호를 만들기 위한 예측 단계(prediction step), 마지막으로 저대역 신호를 만들기 위한 갱신 단계(update step)로 구성되어 있다.

분할 단계(splitting step)는 입력 신호를 두 신호로 분할하는 역할을 수행한다. 이 방법은 필터 뱅크에서는 다상 분해(polyphase decomposition)로 알려져 있고 또한 레이지 웨이블릿 변환 (the Lazy wavelet transform)으로도 불리워 진다. 입력 신호를 x_n이라 하면, 이 입력 신호의 레이지 웨이블릿 변환은 식[4]와 같이 짝수항 신호인 s_k와 홀수항 신호인 d_k로 분할해 주는 역할을 수행한다.

$$s_k = x_{2k},\ d_k = x_{2k+1} \qquad [4]$$

예측 단계(predicting step)는 전 단계인 분할 단계에서 입력 신호 x_n으로부터 얻은 두 신호 s_k와 d_k를 이용하여 고대역 신호를 얻어내는 단계이다. 두 신호 중 한쪽 신호를 적절히 예측함으로써 두 신호간의 상관도를 낮추거나 또는 한쪽 신호의 에너지를 줄일 수 있다.

$$d'_k = d_k - \sum_j P_j s_{k+j} \qquad [5]$$

식[5]에서 P_j는 예측계수가 된다. 잘 선택된 예측계수는 효율적으로 신호 d_k를 예측하여 보다 작은 에너지를 갖는 신호 d'_k값을 제공한다. 예측 방법으로는 주로 보간을 사용한다.

갱신 단계(update step)는 신호 d'_k를 이용해서 신호 s_k가 원 신호 원의 저 대역 신호가 될 수 있게 적절히 처리하는 단계이다. 즉 다른 말로 표현한다면 신호 s_k의 에일리어스 부분을 신호 d'_k를 이용해서 제거해 주는 단계로 해석 할 수 있다. 다음 식에서 에일리어스가 제거된 신호 s'_k를 얻는다.

$$s'_k = s_k + \sum_j u_j d'_{k+j} \qquad [6]$$

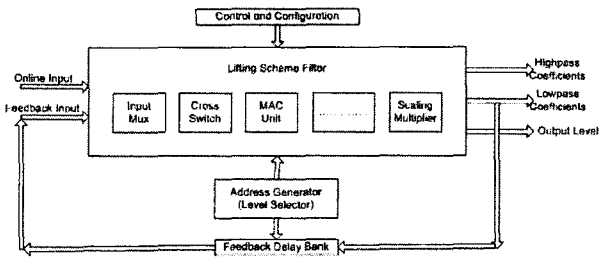
식[6]에서 u_j는 갱신 계수가 된다. 적절한 갱신 계수의 선택은 신호 s_k에 잔존하는 에일리어스 항을 효율

적으로 제거하고 신호 s'_k가 원 입력 신호의 저대역 신호로 되게 한다. 갱신 단계의 적절한 갱신 계수를 선택하기 위해서, 원 입력 신호와 저대역 신호 s'_k의 평균 값이 보존한다는 조건을 이용한다.

분할 단계를 거친 뒤 예측 단계와 갱신 단계를 수행한 후에 얻어지는 저대역 신호와 고대역 신호는 각각 분할된 대역 신호로써 간주할 수 있다. 여기에 부가적인 상관관계가 각 대역별 신호 사이에 존재한다면 예측 단계를 더 수행할 수 있다. 또한 필요하다면 갱신 단계로 한 번 더 수행할 수 있다.

III. 하드웨어 구현

웨이블릿 변환 프로세서를 구현하기 위해서 본 논문에서는 리프팅 필터를 사용한다. 그림 3에서 보이는 바와 같이 리프팅 필터내에는 원래의 입력과 이전 레벨에서 분해된 저대역 계수의 피드백 입력을 선택하는 입력 멀티플렉서와 크로스 스위치, 그리고 실질적으로 계산을 하는 곱셈 누산기(Multiply Accumulate : MAC)와 곱셈기로 구성되어 있다. 다해상도 분해를 위해서는 출력의 저대역 계수를 다시 웨이블릿 분해하기 위한 피드백 지연 부분과 레벨 선택을 위한 어드레스 발생기가 필요하다. 마지막으로 이런 각각의 유닛들이 제대로 연결되기 위한 제어부가 필요하다.



<그림 3. 웨이블릿 프로세서의 구조>

3.1 D4 웨이블릿 필터쌍의 리프팅 분해

Daubechies 4 필터의 h와 g필터는 식[7]로 표현된다.

$$h(z) = h_0 z + h_1 + h_2 z^{-1} + h_3 z^{-2}$$
$$g(z) = h_3 z - h_2 + h_1 z^{-1} - h_0 z^{-2} \quad [7]$$

이때 각각의 FIR 필터 계수는 다음과 같다.

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}},\ h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}},\ h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}},\ h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

필터 쌍은 식[8]과 같이 다상 행렬로 다시 쓸 수 있다.

$$P(z) = \overline{P(z)} = \begin{matrix} h_1 + h_3 z^{-1} & -h_2 - h_0 z^{-1} \\ h_0 z + h_2 & h_3 z + h_1 \end{matrix} \quad [8]$$

인수분해를 하면 식[9]와 같이 두가지 결과를 얻을 수 있다.

$$P(z) = \begin{matrix} 1 & -\frac{1}{\sqrt{3}} z^{-1} \\ 0 & 1 \end{matrix} \begin{matrix} \frac{\sqrt{3}}{4} z + \frac{6-3\sqrt{3}}{4} & 0 \\ 1 & 1 \end{matrix} \begin{matrix} 1 & 0 \\ -\frac{1}{3} & 1 \end{matrix} \begin{matrix} \frac{3+\sqrt{3}}{3\sqrt{2}} & 0 \\ 0 & \frac{3-\sqrt{3}}{\sqrt{2}} \end{matrix}$$

$$P(z) = \begin{matrix} 1 & 0 \\ \frac{1}{\sqrt{3}} z & 1 \end{matrix} \begin{matrix} 1 & -\frac{\sqrt{3}}{4} z^{-1} - \frac{6-3\sqrt{3}}{4} \\ 0 & 1 \end{matrix} \begin{matrix} 1 & 0 \\ \frac{1}{3} & 1 \end{matrix} \begin{matrix} \frac{3-\sqrt{3}}{\sqrt{2}} & 0 \\ 0 & \frac{3+\sqrt{3}}{3\sqrt{2}} \end{matrix} \quad [9]$$

위의 두가지 결과 중에서 식[9]의 결과를 이용하여 식[10]~<식[19]>아래식을 이용해서 웨이블릿 프로세서를 구현한다.

먼저, 행렬로 인수분해 된 것을 그에 적합한 프로세서 구조로 구현하기 위해서는 식을 곱셈과 덧셈의 형로 다시 쓸 필요가 있다.

$$s_n^{(1)} = x_{2n} + \frac{1}{\sqrt{3}} x_{2n-1} \quad [10]$$

$$d_n^{(1)} = x_{2n+1} - \frac{6-3\sqrt{3}}{4} s_n^{(1)} - \frac{\sqrt{3}}{4} s_{n+1}^{(1)} \quad [11]$$

$$s_n^{(2)} = s_n^{(1)} + \frac{1}{3} d_n^{(1)} \quad [12]$$

$$d_n = \frac{3+\sqrt{3}}{3\sqrt{2}} d_n^{(1)} \quad [13]$$

$$s_n = \frac{3-\sqrt{3}}{\sqrt{2}} s_n^{(2)} \quad [14]$$

웨이블릿 프로세서의 출력은 High Pass 출력인 d_n과 Low Pass 출력인 s_n이다. 여기에서 n은 각각의 데이터가 들어오는 이산 시간을 나타낸다.

하나의 신호가 입력되었을 때, 위의 식으로는 이전에 계산되고 저장된 값이 정확히 전달되지 않는다. 이런 문제는 실 생활에 쓰이는 응용분야에서는 자주 있는 일이다. 그래서 이런 문제를 해결하기 위해서 인과관계(causal)가 있는 형태로 바꾸어야 한다.

$$s_n^{(1)} = x_{2n} + \frac{1}{\sqrt{3}} x_{2n-1} = x_{2n} + \alpha_1 x_{2n-1} \quad [15]$$

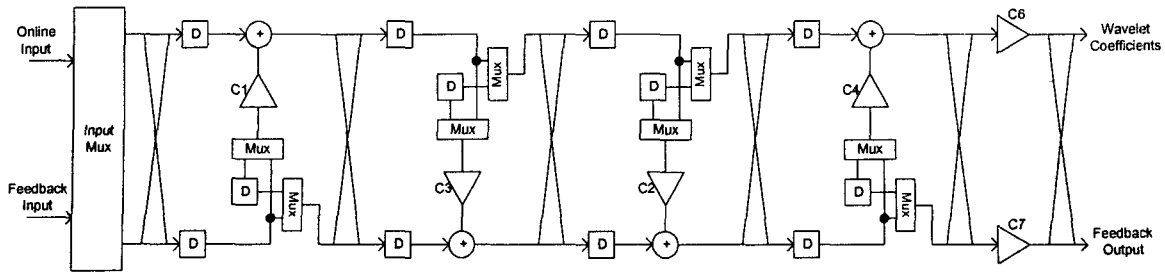
$$d_{n-1}^{(1)} = x_{2n-1} - \frac{6-3\sqrt{3}}{4} s_{n-1}^{(1)} - \frac{\sqrt{3}}{4} s_n^{(1)} = x_{2n-1} + \alpha_2 s_{n-1}^{(1)} + \alpha_3 s_n^{(1)} \quad [16]$$

$$s_{n-1}^{(2)} = s_{n-1}^{(1)} + \frac{1}{3} d_{n-1}^{(1)} = s_{n-1}^{(1)} + \alpha_4 d_{n-1}^{(1)} \quad [17]$$

$$d_{n-1} = \frac{3+\sqrt{3}}{3\sqrt{2}} d_{n-1}^{(2)} = \alpha_5 d_{n-1}^{(2)} \quad [18]$$

$$s_{n-1} = \frac{3-\sqrt{3}}{\sqrt{2}} s_{n-1}^{(2)} = \alpha_6 s_{n-1}^{(2)} \quad [19]$$

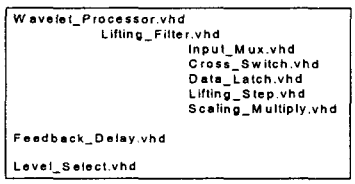
이렇게 필터를 구현하면 그림 4와 같은 리프팅 기반 프로세서를 구현 할 수 있으며 2개의 입력으로 2개의 출력을 계산하는데, 4개의 덧셈기와 6개의 곱셈기, 총 10개의 소자로 구현이 가능하다. 이것은 일반적인 FIR



필터를 구현하는데 필요한 6개의 덧셈기와 8개의 곱셈기, 총 14개의 소자에 비해서 하드웨어 복잡도가 줄어드는 장점이 있다.

3.2 VHDL 모듈 구조

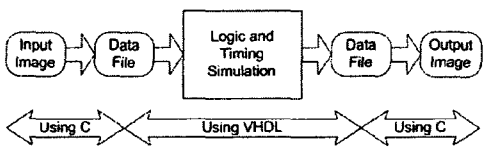
크게 세 부분으로 나누어서 구현하였다. 그림 3에서 보는 것과 같이 리프팅 필터 부분과 어드레스 발생기, 피드백 지연 부분으로 나누어 구현하였다. 각각의 부분을 세부적으로 보면 그림 5의 파일로 구성된다.



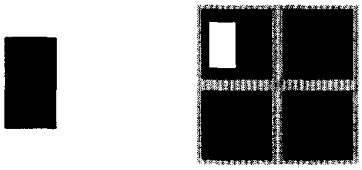
<그림 5. VHDL 파일 리스트>

3.3 검증

그림 6에서 보는 것과 같이 입력 영상을 C언어를 사용해서 데이터 파일로 만든 후에 이것을 웨이블릿 프로세서를 이용해서 시뮬레이션하고 그 결과를 다시 출력 영상으로 만들어서 검증하였다.



<그림 6. 검증 방법>



<그림 7. 단순한 이미지를 이용한 검증 예>

IV. 결론

정지 영상의 전송과 저장을 위해서는 높은 화질을 가지면서 압축률이 뛰어난 영상 압축 기술이 있어야 한다. 그리고 빠른 속도로 영상 처리를 하기 위해서는 하드웨어로 구현해야 한다. 본 논문에서는 JPEG2000에 대응되는 이산 웨이블릿 변환을 VHDL로 구현한다. Daubechies 4 필터를 이용해서 리프팅 기반 웨이블릿 프로세서를 구현 하였으며 2개의 입력으로 2개의 출력을 계산하는데 4개의 덧셈기와 6개의 곱셈기, 총 10개의 소자로 구현이 가능하다. 이것은 일반적인 FIR 필터를 구현하는데 필요한 6개의 덧셈기와 8개의 곱셈기, 총 14개의 소자에 비해서 하드웨어 복잡도가 줄어드는 장점이 있다.

References

[1] G. Fernandez, S. Periaswamy, and W. Sweldens. "LIFTPACK : A software package for wavelet transforms using lifting". Wavelet Applications in Signal and Image Processing IV, pp. 396-408. Proc. SPIE 2825, 1996.

[2] B. Zafarifar. "Micro-coadable discrete wavelet transform". M.Sc. thesis, Delft University of Technology, 1-68340-28(2002)-03

[3] I. Daubechies, W. Sweldens. "Factoring Wavelet Transforms into Lifting Steps". J. Fourier Anal. Appl. Vol. 4, pp. 247-269, 1998

[4] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting based wavelet transform", in Proc. IEEE Workshop Signal Process. Syst., Oct. 2000, pp. 70-79.

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