論文2003-40SD-7-8

Lifting scheme을 이용한 고속 병렬 2D-DWT 하드웨어 구조

(A High Speed 2D-DWT Parallel Hardware Architecture Using the Lifting Scheme)

金 鍾 旭 *, 鄭 正 和 **

(Jong Woog Kim and Jong Wha Chong)

요 약

본 논문은 리프팅 스킴(lifting scheme)의 분할 방법을 개선하여 고속 병렬 처리가 가능한 2차원 DWT(Discrete Wavelet Transform) 하드웨어 구조를 제안 한다. 2차원 DWT 변환은 2차원 입력 데이터 전체에 대하여 연산이 수행되고 순차적으로 2차원 처리가 됨에 따라서 초기 및 전체 지연시간(latency)이 많이 걸린다. 본 논문에서는 처리속도와 지연 시간을 향상시키기 위해 개선된 분할 방법과 새로운 자원 공유 하드웨어 구조를 제안 한다. 상호 연관성이 없는 데이터들을 4 개의 데이터 집합으로 분할하여 병렬 처리에 적합하도록 새로운 분할 방법을 제안하였다. 병렬처리 하드웨어 구조는 하드웨어의 자원 공유가 가능하도록 하기 위해 필터연산의 중간 값을 메모리에 저장할 수 있는 파이프라인 구조를 갖도록 설계하였다. 제안된 구조를 효율적으로 동작 시킬 수 있도록 하드웨어 자원의 공유를 스케줄링하여 초기지연과 전체 지연 시간을 줄였다. 제안하는 구조는 기존의 병렬 처리 구조에 비해 초기 지연 및 전체 지연 시간을 각각 50%와 66%감소시키는 결과를 얻을 수 있었다.

Abstract

In this paper, we present a fast hardware architecture to implement a parallel 2-dimensional discrete wavelet transform(DWT)based on the lifting scheme DWT framework. The conventional 2-D DWT had a long initial and total latencies to get the final 2D transformed coefficients because the DWT used an entire input data set for the transformation and transformed sequentially. The proposed architecture increased the parallel performance at computing the row directional transform using new data splitting method. And, we used the hardware resource sharing architecture for improving the total throughput of 2D DWT. Finally, we proposed a scheduling of hardware resource which is optimized to the proposed hardware architecture and splitting method. Due to the use of the proposed architecture, the parallel computing efficiency is increased. This architecture shows the initial and total latencies are improved by 50% and 66%.

Keyword : DWT, hardware, parallel architecture, wavelet filter

* 正會員, 漢陽大學校 電子工學科

(Dept. of Electronics, Graduate school of Hanyang University)

** 正會員, 漢陽大學校 情報通信大學

(College of Information & Communications, Hanyang University)

接受日字:2002年9月10日, 수정완료일:2003年7月5日

Ⅰ. 서 론

최근 몇 년간 동영상을 비롯한 영상 압축 기법에서 웨이블릿을 이용한 압축 기법들이 소개 되고 있고, MPEG-4와 JPEG2000과 같은 영상 압축 표준안에서 웨이블릿 기반의 압축 기법을 채용하고 있다. 기존의 블록 DCT 기반의 압축 코덱에서는 압축률이 올라감에

따라서 블록의 경계 부분에 심한 열화 현상이 나타나는 블록킹 현상(blocking artifacts)이 나타나는 단점을 가지고 있었다. 그리고, 블록 DCT 기반의 코덱은 다양한 전송 환경에 따라 비례 축소가 가능한(scalability) 코덱을 구현하는데 약점을 가지고 있었다. 이에 반해 웨이블릿은 하나의 프레임을 블록 단위로 분할하지 않고 전체 영상에 대해 변환하기 때문에 블록킹 현상을 줄일 수 있고, 변환의 결과가 웨이블릿의 레벨에 따라 밴드 별로 나누어지기 때문에 전송 레벨의 조정과 양자화 스텝 변화를 통해 화상 크기과 화질에 따라 가변적인 코덱를 구현 할 수 있다는 장점을 가지고 있다.

이러한 웨이블릿 기반의 압축 코덱에 있어서 가장 큰 단점은 전체 영상이 처리 기준이 되기 때문에 2차원의 DWT(Discrete Wavelet Transform)의 최종 결과를 얻기 위한 지연 시간이 길어지고, 중간의 1차원 처리 결과를 저장해 놓기 위한 한 프레임만큼의 버퍼가 필요하다는 단점을 가지고 있다. 이에 다양한 형태의 웨이블릿 기반의 하드웨어 아키텍처가 제안 되고 있는데[1-3, 5-6], 이러한 하드웨어 아키텍쳐들은 초기의 1차원 DWT를 수행하고, 일정 시간 이후에 2 차원 DWT를 수행하는 형태의 구조를 갖고 있다. 기존의 구조들은 1차원 DWT를 처리를 하고, 2차원 DWT의 처리가 시작되기 위해서는 예측과 보상 필터 탭 수의 2배만큼의 행이 처리된 이후에나 열 방향의 2차원 DWT가 시작하게 되어 지연시간이 길어진다. 웨이블릿 하드웨어 구현에 있어서 고려해야 할 점은 이러한 지연시간을 최대한 감소시켜 처리 속도를 향상시키고, 내부 메모리의 크기를 줄이는데 있다고 할 수 있다.

본 논문에서는 처리속도를 향상시키고, 지연시간을 줄이기 위해 리프팅 스킴에서 수행 되는 분할 과정을 상호 연관이 없는 4개의 데이터 집합으로 분할하는 새로운 방법을 제안한다. 분할된 4개의 데이터 집합은 병렬 처리를 통하여 처리속도를 향상 시키게 된다. 병렬 처리에 있어서 하드웨어 자원의 증가를 최소화하기 위해 병렬 처리 하드웨어의 자원 공유가 가능한 구조를 설계하였다. 리프팅 스킴의 웨이블릿 변환에서는 예측과 보상을 위한 필터를 사용하는데, 필터의 특성상 하드웨어 자원을 공유할 때 필터의 지연이 발생한다. 특히 행 방향 연산의 필터의 연산 방향과 행 방향 데이터의 입력 순서가 같아 지연을 줄일 수 있지만, 열 방향 연산의 경우는 열 방향 연산의 방향과 필터의 입력 데이터 순서가 같지 않아 매 연산에서 지연 현상이 발생할 수 있다. 자원의 공유와 효율적인 병렬 처리를 위해서는 하드웨어 자원 할당이 변경 되었을 때 발생하는 연산 지연을 최소화해야 한다. 제안하는 자원 공유 하드웨어는 필터 연산의 중간값을 메모리에 저장하는 방법을 이용하여 연산 지연을 최소화 하였다. 그리고, 자원 공유 하드웨어의 병렬 처리를 위해 자원 할당을 스케쥴링 하였다.

본 논문의 구성은 2장에서 리프팅 스킴의 간략한 구조에 대하여 설명하고, 3장에서는 제안하는 분할방법, 병렬 처리 하드웨어 아키텍처와 지연 시간과 수행 흐름을 설명하였다. 그리고, 최종적으로 4장에서는 기존의 구조와의 결과 비교를 기술하고, 5장에서 결론을 맺는다.

Ⅱ. 리프팅 스킴의 웨이 블릿 구조

일반적인 리프팅스킴의 웨이브릿 변환을 수행하기 위해서는 <그림 1>과 같이 세 단계의 처리 과정을 거치게 되는데, 이것은 각각 입력 데이터에 대한 분할(split), 예측(prediction), 보상(update) 과정이다[4]. 분할은 입력된 데이터를 even/odd의 위치에 따라 두 개의 데이터 집합으로 나누는데, 이 과정은 기존의 웨이블릿에서 행하는 샘플링 과정에 해당하고, 이렇게 분리된 데이터는 고주파 영역의 데이터를 얻기 위한 예측 과정을 거치게 된다. 예측 과정은 분리된 두 개의 데이터 영역의 상관관계를 추출하는 것으로 결국 이미지에서 고주파 영역에 해당하는 데이터를 갖게 된다. 고주파 영역의 데이터를 구한 후에 이미지 전체에 대한 에너지 보존을 위해 보상 과정을 거치게 된다.

보상과정 에 의해 생성된 데이터는 기존의 웨이블릿 방법에서 저주파 영역에 해당하는 데이터 이다.

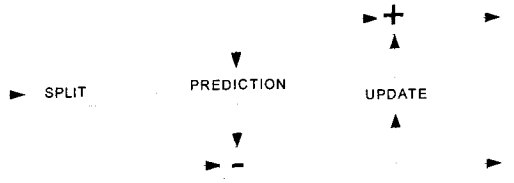


그림 1. 리프팅스킴의 웨이블릿 변환 블록 다이어그램
Fig. 1. Block diagram of lifting scheme wavelet transform.

2차원 리프팅 스킴의 경우 우선 X 방향의 예측과 보

상 과정을 수행하고, 이것을 다시 Y 방향으로 연산을 하여 최종 결과를 얻게 된다. X 방향의 연산을 수식으로 표현 하면 식 (1)과 같다.

$$\begin{aligned} X_H^{new}(n,k) &= x_H(n,k) - P_x(x_L(n,k)) \\ X_L^{new}(n,k) &= x_L(n,k) + U_x(X_H^{new}(n,k)) \\ 0 \le n \le \frac{N}{2} &(N : \text{image width}) \\ 0 \le k \le M &(M : \text{image height}) \end{aligned} \quad (1)$$

여기서 X_L^{new}, X_H^{new} 는 각각 X 방향으로의 예측 과정을 통하여 얻어진 결과 값이고, x_L, x_H 는 입력 데이터를 X 방향으로 분할 한 데이터 집합이다. P(x), U(x)는 각각 예측과 보상 필터를 의미한다.

2차 원의 경우는 식 (1)을 이용하여 X방향으로 변환을 수행하고, 그 결과 값을 다시 Y 방향으로 수행 하여 결과를 얻게 된다. 이러한 과정을 이용한 최종 결과는 4개의 밴드로 구성되는 2차원 변환 결과 값을 갖게 되고 각 밴드별 수행을 위한 수식을 보면 식 (2)와 같이 기술된다.

$$\begin{aligned} X_{HH}^{new}(n,k) &= X_H^{new}(n,2k+1) - P_y(X_H^{new}(n,2k)) \\ X_{HL}^{new}(n,k) &= X_L^{new}(n,2k+1) - P_y(X_L^{new}(n,2k)) \\ X_{LH}^{new}(n,k) &= X_H^{new}(n,2k) + U_y(X_H^{new}(n,2k+1) - P_y(X_H^{new}(n,2k))) \\ X_{LL}^{new}(n,k) &= X_L^{new}(n,2k) + U_y(X_L^{new}(n,2k+1) - P_y(X_L^{new}(n,2k))) \\ 0 \le n \le \frac{N}{2} &(N : \text{image width}) \\ 0 \le k \le \frac{M}{2} &(M : \text{image height}) \end{aligned} \quad (2)$$

Ⅲ. 제안하는 하드웨어 구조

제안하는 구조는 분할 방법, 하드웨어 구조, 스케쥴링 방법으로 구성된다. 분할 방법의 개선은 초기의 병렬 처리 효율을 증가 시켜 초기 지연 시간을 감소시키고, 분할 방법에 적합한 하드웨어 공유 구조 및 전체 병렬 처리 구조와 스케쥴링 방법을 통해 전체 지연시간을 감소 시켜 처리 속도를 향상 시켰다.

1. 제안하는 분할 방법

제안하는 분할 방법은 상호 연관성이 없는 데이터를 4개의 데이터 집합으로 분할하여 초기 병렬 처리 효율을 향상 시킬 수 있도록 하였다. 식 (2)에서 각 첨자들 *n, 2k, 2k+1*는 2차원 데이터의 위치 정보에 해당한다. 이 위치를 정리해 보면 다음의 식 (3)과 같이 첨자를 다시 정리 할 수 있다.

$$\begin{aligned} x_L(n,2k) &\Rightarrow x_{LL}(n,m) \\ x_H(n,k) &\Rightarrow x_{LH}(n,m) \\ x_H(n,k+1) &\Rightarrow x_{HH}(n,m) \\ x_L(n,k+1) &\Rightarrow x_{HL}(n,m) \end{aligned} \quad (3)$$

식 (3)에서 각 밴드 별 입력 데이터 x_{LL}, x_{LH}, x_{HL}, x_{HH}는 <그림 2>와 같이 입력 데이터의 위치에 따라서 4개의 데이터 영역으로 분할 한 것과 같이 된다. 이렇게 함으로써 각 첨자 n, m의 범위를 $0 \le n \le \frac{N}{2}$, $0 \le m \le \frac{M}{2}$ 로 줄일 수 있다.

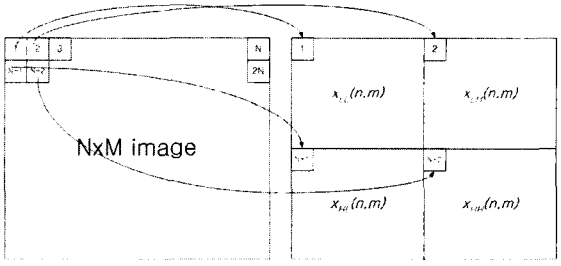


그림 2. 데이터 분할 방법
Fig. 2. The data splitting method.

여기서, X 방향의 DWT에 대하여 식 (1)과 식 (3)을 사용하여 X에 대하여 정리해 보면,

$$X_H^{new\_upper}(n,k) = x_{LH}(n,k) - P_x(x_{LL}(n,k)) \quad (4)$$

$$X_H^{new\_lower}(n,k) = x_{HH}(n,k) - P_x(x_{HL}(n,k)) \quad (5)$$

$$X_L^{new\_upper}(n,k) = x_{LL}(n,k) + U_x(X_H^{new\_upper}(n,k)) \quad (6)$$

$$X_L^{new\_lower}(n,k) = x_{HL}(n,k) + U_x(X_H^{new\_lower}(n,k)) \quad (7)$$

과 같이 전개 할 수 있다. 그 결과 식 (4)와 식 (5)는 동시에 처리함에 있어서 초기에 서로 영향을 주지 않는 x_{LL}, x_{HL} 데이터 집합을 사용하기 때문에 두개의 예측 필터를 이용하여 병렬로 처리할 수 있고, 식 (6)과 식 (7)역시 두개의 보상 필터를 이용하여 병렬로 처 리 할 수 있게 된다.

2차원 DWT의 결과는 1차원 결과를 이용하여 처리하게 되는데, 이것을 수식으로 표현하면 다음의 식 (8) ~식 (11)와 같다.

$$X_{LH}^{new}(n,k) = X_{H}^{new\_upper} + U_y(X_{HH}^{new}(n,k)) \quad (8)$$

$$X_{HH}^{new}(n,k) = X_{H}^{new\_lower} - P_y(X_{H}^{new\_upper}(n,k)) \quad (9)$$

$$X_{LL}^{new}(n,k) = X_{L}^{new\_upper} + U_y(X_{HL}^{new}(n,k)) \quad (10)$$

$$X_{HL}^{new}(n,k) = X_{L}^{new\_lower} - P_y(X_{L}^{new\_upper}(n,k)) \quad (11)$$

제안하는 구조에서는 4개의 데이터 분할 영역에 의해 1차원 DWT의 결과 값 $X_L^{new\_upper}$, $X_H^{new\_upper}$, $X_L^{new\_lower}$, $X_H^{new_lower}$을 다시 병렬 처리를 수행 할 수 있다. 그 결과 식 (8)~식 (11)의 결과 계산을 빨리 수행함에 따라서 전체 지연 시간을 줄일 수 있다.

2. 제안하는 병렬 처리 하드웨어 구조

일반적인 순차처리 구조의 웨이블릿 하드웨어는 <그림 3(a)>와 같은 처리 순서를 갖게 된다. 이러한 순차 구조는 구현에 있어서 하드웨어 복잡도는 낮지만 1차원의 변환을 순차적으로 수행하기 때문에 최종 2차원 결과를 얻기 위해서는 1차원 처리가 모두 수행 될 때까지 기다려야 하는 단점을 가지고 있다. 이러한 단점으로 인해 지연 시간이 많이 걸리고, 1차원의 처리결과를 2차원 처리를 위해 모두 저장해 놓아야 하기 때문에 중간의 프레임 버퍼의 크기가 전체데이터만큼 필요하게 된다. 본 논문에서 제안하는 방법은 <그림 3(b)>

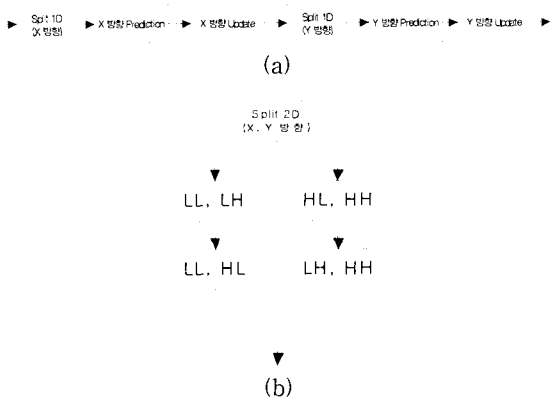


그림 3. 2차원 웨이블릿 처리 과정 (a) 기존의 2차원 웨이블릿 처리 과정, (b) 제안하는 2차원 웨이블릿 처리 과정

Fig. 3. 2D Wavelet transform procedure. (a) The conventional sequential wavelet transform procedure (b) Proposed wavelet transform procedure.

와 같이 식 (4)~식 (7) 수식을 바탕으로 초기의 병렬 처리를 효율을 향상시켜 초기 지연을 줄이도록 하였다. 지연시간을 줄이므로 해서 2차원의 결과를 빨리 얻을 수 있고, 그 결과 1차원 데이터 값을 저장하는 버퍼의 크기를 줄이는 장점을 갖게 된다.

기존의 구조에서는 초기 분할을 행 방향으로만 하여 처리함으로 예측과 보상만을 병렬로 처리 할 수 있었다. 즉 예측을 위한 모듈과 보상을 위한 모듈 한 개씩을 사용하게 된다. 이러한 구조를 이용 할 경우 1차원의 예측과 보상은 병렬로 처리 할 수 있지만, 2차원 처리를 시작하기 위한 초기 지연 시간은 줄어들지 않은 상태가 된다. 본 구조에서는 1차원 웨이블릿의 처리 속도를 향상시키기 위해 <그림 4>와 같이 2개씩의 예측과 보상 모듈을 사용한다. 이러한 구성은 4개의 데이터 집합으로 입력 데이터를 분할함으로써, 상위 2개의 분할 영역(x_{LL}, x_{LH})과 하위 2개의 분할영역(x_{HL}, x_{HH})에 대하여 병렬로 처리 할 수 있게 된다. 이렇게 상위 영역과 하 위 영역을 병렬로 처리함으로써, 2차원 처리에 필요한 1차원 결과 값을 빨리 얻을 수 있게 된다. 그리고 2차원의 연산 시작을 빨리 하기 위해 P3, U3의 모듈을 사용하여 2차원 처리 지연 시간을 단축 시켰다.

<그림 4>에서 $x_{LL}(n,m)$, $x_{LH}(n,m)$ 데이터는 P1, U1 모듈을 통하여 연산의 결과를 버퍼 메모리에 저장하고, $x_{HL}(n,m)$, $x_{HH}(n,m)$ 데이터는 P2, U2를 이용하여 동시에 처리 하도록 1차원 DWT를 구성하였다. 이렇게 함으로써 1차원의 결과를 얻기 위한 처리 속도를 2배로 향상 시킬 수 있다. 이렇게 병렬 처리된 1차원 처리 결과는 <그림 5>에서 처럼 2차원 변환을 위한 입력 데이터로사용 되게 된다. 제안하는 구조 에서는 이러한 1차원의 결과를 저장하는데 있어서 1프레임 분량의 메모리가 필요한 것이 아니고, 4LhN(Lh : prediction 필터의 탭수, N : 이미지의 가로 픽셀수) 만큼의 임시 버퍼만을 이용하여 2차원 DWT 를 수행 할 수 있다. 이것은 <그림 5>에서와 같이 열 방향의 변환 과정을 수행하기 위해서는 행 방향의 L_h 개 행만큼의 데이터가 필요로 하게 된다. 이렇게 L_h만큼의 데이터가 처리가 되면 행 방향의 변환 과정이 수행되게 된다.

4-탭의 예측 필터를 사용하는 경우를 예로 들어 동작은 기술하면 다음 의 식 (9)와 같다. 식 (9)에서 입력 데이터 R[i]는 계수 a_n과의 연산을 통하여 내부 메모리에 저장 되게 되는데, 최종 결과를 얻기 위해서는 초기에 필터 탭 수만큼의 초기 지연이 있고, 그 이후에는

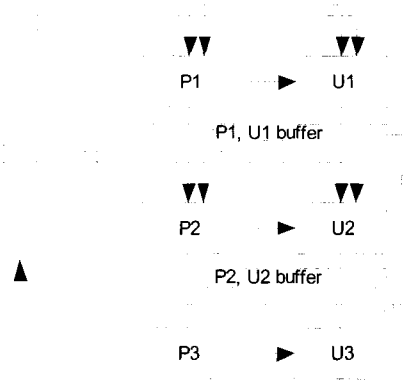


그림 4. 2D-DWT를 위한 구조의 블록 다이어그램
Fig. 4. Functional partition of 2-D DWT decomposition with lifting scheme.

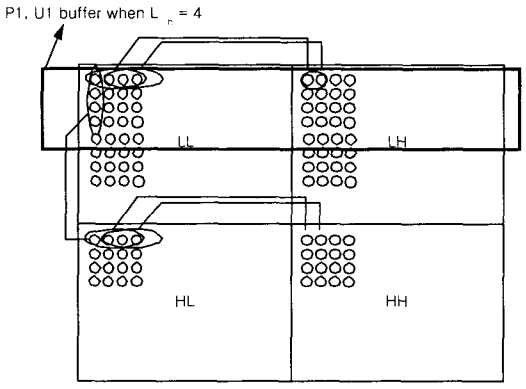


그림 5. 2차원 데이터 연관성 및 버퍼 메모리의 저장 범위
Fig. 5. 2-D data dependency and buffer memory covered area

최종 결과가 연속적으로 나오게 된다. 이러한 구조는 기존의 필터들이 하나의 결과 값을 구하기 위하여 필터 계수와 입력 데이터의 연산을 동시에 수행하던 것을 파이프라인 구조로 바꾸고, 기존의 필터에서 레지스터만을 사용하는 구조를 레지스터와 내 부 메모리를 함께 사용하였다. 이렇게 사용함으로써, 자원 공유를 위한 스케쥴링 변화 시에 발생하는 데이터 지연과 덧셈기 의 입력을 2입력 덧셈기로 통일할 수 있는 구조적 효과를 얻을 수 있다.

$$
\begin{aligned}
t_0 &: M[i] \leftarrow a_0 R[i] \\
t_1 &: M[i+1] \leftarrow M[i] + a_1 R[i+1] \\
&\quad M[i] \leftarrow a_0 R[i+1] \\
t_2 &: M[i+2] \leftarrow M[i+1] + a_2 R[i+2] \\
&\quad M[i+1] \leftarrow M[i] + a_1 R[i+2] \\
&\quad M[i] \leftarrow a_0 R[i+2] \\
t_3 &: M[i+3] \leftarrow M[i+2] + a_3 R[i+3] \\
&\quad M[i+2] \leftarrow M[i+1] + a_2 R[i+3] \\
&\quad M[i+1] \leftarrow M[i] + a_1 R[i+3] \\
&\quad M[i] \leftarrow a_0 R[i+3]
\end{aligned}
\tag{9}
$$

<그림 6>와 <그림 7>은 각각 P, U 모듈 의 구조를 나타낸 것으로 각 모듈은 필터 탭 수만큼의 곱셈기와 덧셈기로 구성되며, 메모리와 연산의 중간 값들이 메모리에 보관 되는 구조를 갖게 된다.

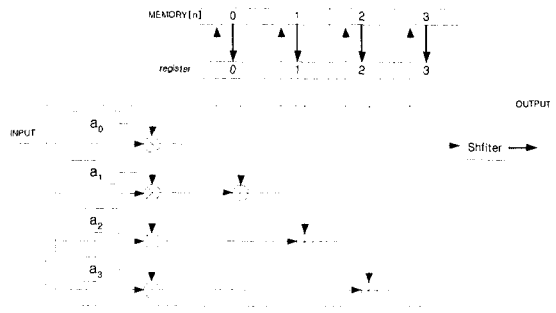


그림 6. P 모듈의 구조 및 데이터 흐름도(4-탭 필터)
Fig. 6. P module architecture and signal flow(4-tap filter)

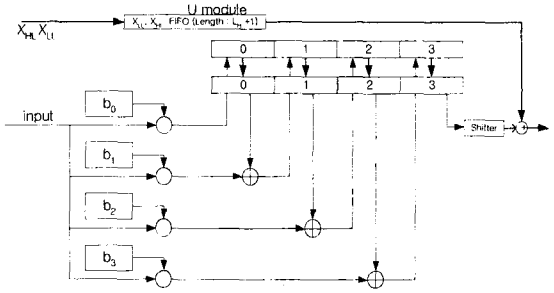


그림 7. U 모듈의 구조 및 데이터 흐름도(4-탭 필터)
Fig. 7. U module architecture and signal flow(4-tap filter).

<그림 7>의 U 모듈에 있는 FIFO는 예 측필터의 탭 수와 보상 필터의 탭 수가 같지 않을 때 발생하는 X_{LL}, X_{HL} 데이터의 지연 차이를 보상해 주기 위한 FIFO 이다.

3. 스케쥴링 및 지연시간

기본 적인 2차원 DWT 구조에서 순차 구조를 갖고 행과 열의 개수가 같은 경우에는 최초 2차원 처리 결

과를 얻기 위하여 $\frac{N^2}{2}+L_h+N_l$ (N: 열의 개수, 행의 개수) 만큼의 처리 지연 시간이 걸리고, 전체 2차원 DWT를 완료 하는 데는 $O(N^2)$ 만큼의 지연을 갖게 된다. 통상의 병렬 처리 구조에서는 이러한 지연을 줄이기 위해 1차원 결과를 파이프라인을 통하여 처리하는데 이러한 경우라도, 1차원 예측 필터의 탭 수만큼의 지연($2NL_h+L_l$)이 발생하게 된다. 본 구조에서는 이러한 1차원 지연을 최소화하기 위하여 P1, P2, U1, U2의 두 개의 1차원 처리 모듈을 사용하였고, P3, U3 모듈을 2차원 처리에 이용하여 최종 지연 시간을 감소 시켰다.

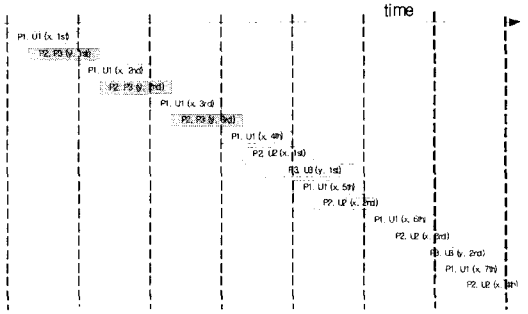


그림 8. 초기 동작 스케줄링
Fig. 8. The initial state schedule.

초기의 동작은 <그림 8>과 같이 진행 되는데, 우선 P1 모듈이 동작하여 첫 번째 라인의 데이터에 대한 1차원 DWT를 진행 하고, L_h개만큼의 데이터가 처리되면 그 값은 Y 방향의 prediction에 사용 할 수 있는 상황이 된다. 이것을 P3 모듈을 이용하여 계산하고, U1이 시작되고, update된 데이터 중에 U1의 결과는 Y 방향의 DWT에서는 prediction에 사용 되는 데이터이기 때문에 U1의 최종 결과가 나오는 시점에서부터는 Y 방향의 prediction을 시작할 수 있다. 그러나 P3가 P1의 결과를 처리하는 도중에 있기 때문에 아직 동작을 시작하지 않고 있는 P2를 이용하여 처리하게 된다.

X 방향의 DWT에서 초기에 P2 모듈은 P1 모듈보다 늦게 시작하게 되는데, 그 이유는 P2의 결과를 이용하여 Y 방향의 처리를 할 때, 우선 X 방향의 prediction 결과가 나와 있어야 처리가 가능하다. 즉 P1 모듈이 L_h 행만큼을 처리 했을 때부터 P2의 1 차원 결과를 활용하게 된다. 그래서 P1이 L_h 행만큼을 처리할 때 까지는 Y 방향의 prediction에 공유하여 사용할 수 있게 된 다. 이렇게 함으로써 Y 방향의 병목 현상을 해소 하게 된다. <그림 8>은 초기 동작 스케줄을 나타내고 있다.

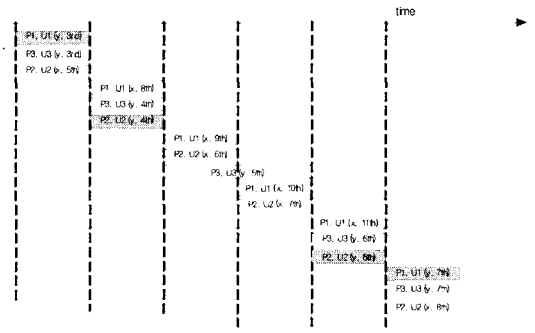


그림 9. 초기 동작 이후의 동작 스케줄링
Fig. 9. The steady state processor schedule after the initial state.

초기 지연 이후에는 <그림 9>와 같이 동작하게 되는데, X 방향의 DWT를 처리하는 모듈은 2개씩 존재 하지만 Y 방향을 처리하는 모듈 은 1개씩만 존재 하게 된다. 이런 경우 1차원 이 빨리 끝나기 때문에 2차원 처리에서 병목 현상이 발생하여 버퍼 메모리가 1프레임의 크기만큼이 필요하다. 하지만 제안하는 구조에서는 이러한 병목 현상을 줄이기 위하여 P1, P2 모듈을 번갈아 가면서 Y 방향에 연산에 할당하여 이러한 병목 현상을 줄이고, 전체 처리량(throughput)을 향상시키게 된다.

즉 완전한 2-D 분 할 구조에서는 X, Y 방향 모두 2개씩의 P, U모듈을 사용하게 되는데, 이런 경우 전체 처리량은 $\frac{1}{4}$로 줄지만 이렇게 할 경우에는 내부 버퍼 메모리와 하드웨어 코스트 모두가 증가 하게 된다.

그래서, X 방향의 P, U 모듈(P1, P2, U1, U2)을 Y 방향에서 공유를 하게 되면, 전체 처리량은 $\frac{1}{3}$로 줄지만 하드웨어 증가를 50%로 줄일 수 있다.

병렬 구조를 사용하지 않았을 때 초기 지연은 $N^2+(L_h\times N)+L_l$이다. 이러한 초기 지연은 X방향 DWT가 끝난 이후에 Y 방향의 처리가 시작되기 때문이다. 이러한 초기 지연을 줄이기 위해 병렬 구조가 사용되고 있는데, 제안하는 구조에서는 초기에 $\frac{L_h\times N}{2}+L_l$로 줄 일수 있다. 이러한 지연의 감소는 2-D 분할을 사용하지 않는 다른 병렬 구조에서의 $(L_h\times N)+L_l$ 에 비하여 절반으로 줄이는 효과를 얻을 수 있다.

Ⅳ. 결과 및 고찰

제안하는 구조는 기존의 병렬 처리 구조에 비하여 하드웨어는 증가 하였지만 시스톨릭 어레이를 이용한 구조에 비하여서는 감소된 하드웨어 구조를 얻을 수 있었다. 그리고, 제안하는 구조는 새로운 분할 방법을 이용한 병렬 처리 효율을 증가시켜 내부 메모리 버퍼의 용량을 감소 시켰으며, 초기 지연 시간을 줄일 수 있도록 1차원 DWT의 병렬 처리를 증가 시켰다. 그 결과 초기 지연시간을 50% 정도 감소시킬 수 있었다. <표 1>은 기존의 구조와 제안하는 구조와의 멀티플라이어 수와 메모리 사이즈, 그리고, 지연시간에 대한 비교표이다. <표 1>에서 알 수 있듯이 하드웨어 자원의 가장 큰 사이즈를 점유하고 있는 멀티플라이어의 수가 기존의 병렬 처리 방법에 비하여 50% 정도 증가 하였다. 하지만 내부 메모리 버퍼 사이즈는 감소하였고 초기 병렬 처리 속도 향상과 자원 공유를 통한 2차원 DWT의 병렬 처리로 인해 전체 지연 시간을 기존의 방법의 1/3로 줄일 수 있었다.

표 1. 기존 구조와 멀티플라이 어수, 내부 메모리 크기, 초기 지연 시간 및 전체 지연시간의 비교표

Table 1. The comparison of the multiplication resource, memory size, initial latency and total latency to the previous architecture.

Resource	Systolic method[5]	Previous Lifting[6]	Proposed
Multipliers	$8L_h+4$	$2L_l + 2L_h$	$3L_l + 3L_h$
Memory Size	$8NL_h+4N$	$2N(2L_h 1+ L_l)$	$4NL_h$
Inital latency	$(L_h\times N)+L_l$	$(L_h\times N)+L_l$	$\frac{L_h \times N}{2} + L_l$
Total latency	$N^2+a\ (a \ll N^2)$	$N^2+\beta\ (\beta \ll N^2)$	$\frac{N^2}{3}+\gamma(\gamma \ll \frac{N^2}{3})$

Ⅴ. 결 론

본 논문에서는 새로운 데이터 분할 방법과 자원 공유 하드웨어를 이용하여 리프팅 스킴의 2D-DWT의 지연시간과 처리 속도를 향상 시키는 하드웨어 구조를 제안하였다. 웨이블릿 변환기는 초기지연과 전체 처리 지연의 단점을 가지고 있었고, 이러한 단점을 해결하기 위해서는 초기의 병렬 처리 효율을 증대 시키고, 2D-DWT의 최종 데이터를 얻기까지의 지연시간을 줄이는 것이 필요하다.

제안하는 구조는 기존의 구조들과는 다른 분할 방법을 이용하여 초기의 행 방향 연산을 병렬 처리하여 처리 속도를 향상 시켰다. 행 방향 연산의 속도 향상은 열 방향 연산의 시작 시간을 앞당겨 지연시간을 줄일 수 있었다. 그리고, 필터의 중간 연산 결과를 버퍼 메모리에 저장하는 파이프라인 구조의 자원 공유 하드웨어를 통해 속도 향상과 하드웨어 사이즈 증가를 억제할 수 있었다.

실험결과 기존의 병렬 처리구조에 비해 하드웨어 사이즈는 50% 정도 증가 하였지만 전체 처리 속도를 66% 정도로 향상 시킬 수 있었고, 초기 지연을 기존의 방법에 비하여 50% 정도 개선한 결과를 얻을 수 있었다.

참 고 문 헌

[1] Wenqing Jiang, and Antonio Ortega, "Lifting Factorization-Based Discrete Wavelet Transform Architecture Design," IEEE Trans. on Circuits and System for Video Technology, vol. 11, no. 5, pp. 651~657, May 2001.

[2] Andra, K., Chakrabarti, C., and Acharya, T., "A VLSI architecture for lifting-based wavelet transform," IEEE Workshop on Signal Processing Systems 2000, pp. 70~79, 2000.

[3] Andra, K., Chakrabarti, C., and Acharya, T., "Efficient implementation of a set of lifting based wavelet filters," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 1101~1104, 2001.

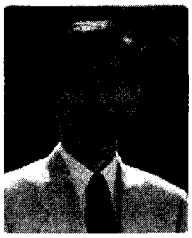
[4] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," Processings of SPIE, vol. 2569, pp. 68~79, 1995.

[5] M. Vishwanath, R.M. Owens, and M.N. Irwin, "VLSI Architecture for the Discrete Wavelet Transform," IEEE Trans. on Circuits and Systems-II : Analog and Digital Signal Processing, vol. 42, No. 5, pp. 305~316, 1995.

[6] M. Ferretti, and D. Rizzo, "A Parallel

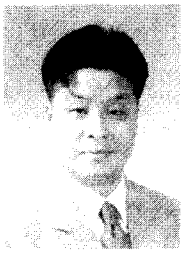
Architecture for the 2-D Discrete Wavelet Transform with Integer Lifting Scheme," Journal of VLSI Signal Processing, vol. 28, pp. 165~185, 2001.

저 자 소 개



鄭 正 和(正會員)

1975년 2월 : 한양대학교 전자공학과(공학사). 1977년 2월 : 한양대학교 대학원 전자공학과(석사). 1981년 3월 : 일본 와세다대학교 대학원 전자공학과(박사). 1986년 6월~1987년 9월 : 미국 Berkeley 대학 박사후 과정. 현재 : 한양대학교 정보통신 대학 교수 재직. <주관심분야 : HW/SW Co-design, High Speed wireless LAN system, MPEG encoder/decoder chip design>



金 鍾 旭(正會員)

1992년 2월 : 한양대학교 전자공학과(공학사). 1994년 2월 : 한양대학교 전자공학과(석사). 현재 : 한양대학교 대학원 전자공학과 박사과정. <주요관심분야 : MPEG-4 video codec 및 hardware architecture, wavelet transform 기반의 video codec.>