論文2003-40CI-2-3

실시간 내장형 응용을 위한 2차원 웨이브렛 변환 프로세서

(2D DWT Processor for Real-time Embedded Applications)

丁 甲 天 *, 朴 性 模 **

(Gab Cheon Jung and Seong Mo Park)

요 약

본 논문에서는 상태 변수 표현 방법에 따른 알고리즘 분할을 통해 2차원 웨이브렛 변환 연산을 실시간으로 처리할 수 있는 프로세서 구조를 제안하였다. 제안된 프로세서 구조는 영상입력에 대하여 행, 열 방향을 동시에 고려하여 데이터 플로우 방식으로 처리함으로써 중간적인 결과의 메모리 저장 및 읽기에 소요되는 전달 지연 시간을 감소할 수 있어 실시간 처리에 적합한 VLSI 구조이다. 필터의 길이를 K라할 때 프로세서는 내부에 4개의 곱셈기, 4개의 덧셈기 및 NK-N 크기의 메모리를 가지는 등의 하드웨어 복잡도가 낮아 웹 카메라 서버와 같은 내장형의 응용에 매우 적합한 구조이고, 쉽게 어레이 구조로 확장할 수 있어 고성능을 요구하는 다양한 영상 처리 응용에도 사용 가능하다.

Abstract

In this paper, a processor architecture is proposed based on the state space implementation technique for real time processing of 2-D discrete wavelet transform(DWT). It conducts 2-D DWT operations in consideration of row and column direction simultaneously, thus can reduce latency due to memory access for storing intermediate results. It is a VLSI architecture suitable for real time processing. The proposed architecture includes only four multipliers and four adders, and NK-N internal memory storage, where K denotes the length of filter. It has a small hardware complexity. Therefore it is very suitable architecture for real time, embedded applications such as web camera server. Since the processor is easily extended to array structure, it can be applied to various image processing applications.

Keywords: 2-D DWT, state space, processor architecture, hardware complexity, real time

1. 서 론

웨이브렛 변환(Wavelet Transform)은 영상처리(패턴 인식, 에지추출), 영상 압축, 신호분석 등에 널리 사용되고 있으며, 특히 2차원 이산 웨이브렛 변환(DWT: Discrete Wavelet Transform)의 실시간 구현은 많은 이미지 처리 응용들에 요구되어진다. 이산 웨이브렛 변환은 분해를 위한 필터링 및 다운샘플링, 합성을 위한 필터링 및 보간 등의 복잡한 연산을 필요로 하며, 특히 2차원 웨이브렛 변환의 경우 행 방향과 열 방향을 모

* 正會員, 全南大學校 電子工學科
(Dept, of Electronics Eng., Chonnam National Univ.)
** 正會員, 全南大學校 컴퓨터工學科
(Dept, of Computer Eng., Chonnam Natinal Univ.)

※ 본 논문은 한국과학재단 지정 전남대학교 고품질 전기전자부품 및 시스템연구센터의 연구비 지원에 의해 연구되었음.
※ This work was supported in part by IDEC.
接受日字:2002年7月24日, 수정완료일:2003年3月5日

두 처리해야 하므로 연산량이 많아 실제적인 응용을 위해서는 특수한 하드웨어의 개발이 요구되어진다. 2차원 이산 웨이브렛 변환에 대해 여러 하드웨어 구조들이 제안되었으며, VLSI 칩으로 구현되었다[1-6].

A. Grzeszczak, M. K. Mandal, S. Panchanathan[1]의 구조는 필터 셀당 하나의 곱셈기를 사용한 1-D 시스톨릭 어레이 구조로서, 간단하고 작은 칩 면적을 차지하며 다차원 DWT에 쉽게 확장 가능하다. 그러나 2-D DWT의 경우 행 열 분해에 따른 한 방향 연산 후의 중간적인 결과들을 저장하는데 거대한 량의 전치(transpose) 메모리가 필요하며 메모리 접근에 따른 전달 지연도 길어지는 단점이 있다. 이와 같은 행 열 분해 방법의 단점을 극복하기위해 제안된C. Chakrabarti, M. Vishwanath[2]의 병렬 필터 구조와 Chu Yu와 Sao-Jie Chen[3]의 병렬 시스톨릭 구조는 non-separable접근 방식을 이용하여 행방향과 열방향을 동시에 처리하도록 함으로써 빠른 연산 속도를 가지나 내부에 곱셈기, 덧셈기, 내부 메모리 등의 리소스가 많이 요구되어 내장형의 응용에는 적합하지가 않다. 이밖에도 웨이브렛 연산을 컨볼루션(convolution)을 사용하지 않고 리프팅(lifting) 방법으로 구현 한 구조도 제시되었다[6].

본 논문에서는 separable접근 방식이면서 2차원 이산 웨이브렛 변환을 실시간으로 구현할 수 있는 데이터 플로우 기반 프로세서 구조를 제안하였다. Ⅱ장에서 상태 변수 표현법에 따른 알고리즘 분할에 대하여 기술하고, Ⅲ장에서는 알고리즘 분할에 따른 웨이브렛 변환 프로세서 구조에 대하여 기술하였다. 그리고 Ⅳ장에서는 프로세서의 VHDL 모델링 및 검증과 다른 2차원 웨이브렛 변환 VLSI 구조들과의 비교 및 성능 분석을 기술하며, Ⅴ장에서 결론을 맺었다.

Ⅱ. 알고리즘 분할

2차원 웨이브렛 변환은 행방향으로의 1차원 웨이브렛 변환과 열방향으로의 1차원 웨이브렛 변환으로 생각할 수 있다. 따라서 2차원 이산 웨이브렛 변환은 영상의 행과 열을 1차원 필터들로 순차적으로 연이어 필터링함으로써 계산할 수 있다. 이것은 separable conjugate mirror 필터 분해와 같은 알고리즘으로 2차원 product separable 필터 뱅크의 피라미드 구조로 볼 수 있다. 2차원 product separable 필터는 식 (1)과 같이 나타낼 수 있다.

$$y(n_1,n_2)=\sum_{i=0}^{N_2}\sum_{j=0}^{N_1}h(i,j)x(n_1-i,n_2-j)=\sum_{i=0}^{N_2}h(i)\sum_{j=0}^{N_1}h(j)x(n_1-i,n_2-j) \quad (1)$$

여기서 h(i,j)는 필터의 임펄스 응답을 나타내고, N_1, N_2는 수평 수직 방향의 필터의 차수를 나타낸다. 식 (1)의 두 번째 식은 product separable 2차원 필터의 임펄스 응답인 h(i,j)가 1차원 필터의 임펄스 응답인 h(i)와 h(j)의 곱으로 분리되는 것을 보여주며, 또한 2차원 필터링 알고리즘의 행-열 분할(row-column decomposition)을 보여주고 있다. 이와 같이 분할된 알고리즘은 상태 변수 표현(state space representation) 방법을 사용하여 효율적으로 구현할 수 있으며, 식 (1)의 2차원 product separable 필터는 수직 상태 변수들($q_{2,1}$, $q_{2,2}$, $q_{2,3}$, ... , $q_{2,N2-1}$)과 수평 상태 변수들($q_{1,1}$, $q_{1,2}$, $q_{1,3}$, ... , $q_{1,N1-1}$)을 사용하여 식 (2), (3)에 나타낸 바와 같이 상태 변수 방정식들의 집합으로 분할할 수 있다[7].

$$\begin{aligned}
y_0(n_1,n_2) &= v(0)x(n_1,n_2)+q_{2,1}(n_1-1,n_2)\\
q_{2,1}(n_1,n_2) &= v(1)x(n_1,n_2)+q_{2,2}(n_1-1,n_2)\\
q_{2,2}(n_1,n_2) &= v(2)x(n_1,n_2)+q_{2,3}(n_1-1,n_2)\\
&\vdots\\
q_{2,N_2-2}(n_1,n_2) &= v(N_2-2)x(n_1,n_2)+q_{2,N_2-1}(n_1-1,n_2)\\
q_{2,N_2-1}(n_1,n_2) &= v(N_2-1)x(n_1,n_2)
\end{aligned} \quad (2)$$

$$\begin{aligned}
y(n_1,n_2) &= h(0)y_0(n_1,n_2)+q_{1,1}(n_1,n_2-1)\\
q_{1,1}(n_1,n_2) &= h(1)y_0(n_1,n_2)+q_{1,2}(n_1,n_2-1)\\
q_{1,2}(n_1,n_2) &= h(2)y_0(n_1,n_2)+q_{1,3}(n_1,n_2-1)\\
&\vdots\\
q_{1,N_1-2}(n_1,n_2) &= h(N_1-2)y_0(n_1,n_2)+q_{1,N_1-1}(n_1,n_2-1)\\
q_{1,N_1-1}(n_1,n_2) &= h(N_1-1)y_0(n_1,n_2)
\end{aligned} \quad (3)$$

식 (2)는 입력 $x(n_1,n_2)$와 이전 수직 상태 변수들 $q_{2,i}(n_1-1,n_2)$를 이용하여 수직 방향으로의 열 연산을 수행한다. 식 (3)은 식 (2)의 $y_0(n_1,n_2)$와 이전 수평 상태 변수들 $q_{1,j}(n_1,n_2-1)$을 이용하여 수평 방향으로의 행 연산을 수행한다. 위의 식에서 v(0), v(1), ….v(N_2-1)는 식 (1)에서의 h(i)를, h(0), h(1), …h(N_1-1)는 식 (1)의 h(j)를 의미한다.

웨이브렛 변환의 다운샘플링 특성에 따라 짝수 번째 픽셀(행방향, 열방향)에서의 연산 결과는 사용되지 않

으므로 식 (2), (3)은 식 (4), (5)와 같이 변환 할 수 있어 연산량과 필요한 상태 변수들의 수를 절반으로 감소 시킬 수 있다. <그림 1>은 식 (4), (5)에 따른 영상의 픽셀들에 대한 데이터 의존성을 나타내며, 그림에서 V.S는 수직 상태변수를 H.S는 수평 상태변수를 나타낸다.

$$y_0(2n_1,n_2)=v(0)x(2n_1,n_2)+v(1)x(2n_1-1,n_2)+q_{2,1}(2n_1-2,n_2) \quad (4)$$
$$q_{2,1}(2n_1,n_2)=v(2)x(2n_1,n_2)+v(3)x(2n_1-1,n_2)+q_{2,2}(2n_1-2,n_2)$$
$$q_{2,2}(2n_1,n_2)=v(4)x(2n_1,n_2)+v(5)x(2n_1-1,n_2)+q_{2,3}(2n_1-2,n_2)$$
$$\vdots$$
$$q_{2,i-1}(2n_1,n_2)=$$
$$v(N_2-3)x(2n_1,n_2)+v(N_2-2)x(2n_1-1,n_2)+q_{2,i}(2n_1-2,n_2) \quad if N_2=even$$
$$v(N_2-4)x(2n_1,n_2)+v(N_2-3)x(2n_1-1,n_2)+q_{2,i}(2n_1-2,n_2) \quad if N_2=odd$$
$$q_{2,i}(2n_1,n_2)=$$
$$v(N_2-1)x(2n_1,n_2) \quad if N_2=even$$
$$v(N_2-2)x(2n_1,n_2)+v(N_2-1)x(2n_1-1,n_2) \quad if N_2=odd$$

, for $n_1 = 0,1,\cdots,(N/2)-1$, $n_2 = 0,1,\cdots,N-1$, $i = 1,\cdots,N_2/2$

$$y(n_1,2n_2)=h(0)y_0(n_1,2n_2)+h(1)y_0(n_1,2n_2-1)+q_{1,1}(n_1,2n_2-2) \quad (5)$$
$$q_{1,1}(n_1,2n_2)=h(2)y_0(n_1,2n_2)+h(3)y_0(n_1,2n_2-1)+q_{1,2}(n_1,2n_2-2)$$
$$q_{1,2}(n_1,2n_2)=h(4)y_0(n_1,2n_2)+h(5)y_0(n_1,2n_2-1)+q_{1,3}(n_1,2n_2-2)$$
$$\vdots$$
$$q_{1,j-1}(n_1,2n_2)=$$
$$h(N_1-3)y_0(n_1,2n_2)+h(N_1-2)y_0(n_1,2n_2-1)+q_{1,j}(n_1,2n_2-2) \quad if N=even$$
$$h(N_1-4)y_0(n_1,2n_2)+h(N_1-3)y_0(n_1,2n_2-1)+q_{1,j}(n_1,2n_2-2) \quad if N=odd$$
$$q_{1,j}(2n_1,2n_2)=$$
$$h(N_1-1)y_0(n_1,2n_2) \quad if N=even$$
$$h(N_1-2)y_0(n_1,2n_2)+h(N_1-1)y_0(n_1,2n_2-1) \quad if N=odd$$

, for $n_1 = 0,1,\cdots,(N/2)-1$, $n_2 = 0,1,\cdots,(N/2)-1$, $j = 1,\cdots,N_1/2$

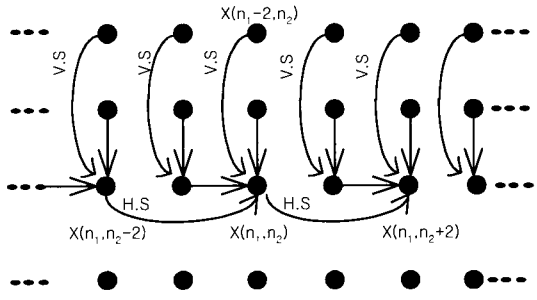


그림 1. 알고리즘 분할에 따른 데이터 의존성
Fig. 1. Data dependency according to the algorithm decomposition.

Ⅲ. 프로세서 구조

1. 제안된 프로세서 구조

제안된 웨이브렛 변환 프로세서는 수직 프로세서와 수평 프로세서의 두 개의 프로세서로 구성된다. 수직 프로세서는 영상의 열들을 필터링하고 수평 프로세서는 영상의 행들을 필터링한다. 수직 프로세서는 한 행의 영상 데이터를 받아서 만약 짝수번째 행 이면 연산을 수행하지 않고 하나의 행으로 구성된 내부의 라인 메모리에 저장한다. 홀수번째 행이면 식 (4)에 나타낸 열방향으로의 상태 변수 방정식을 수행하여 출력 y_0와 상태 변수 값들을 계산한다. 수직 프로세서는 필터링된 출력 y_0를 수평 프로세서에게로 보내고, 계산된 수직 상태 변수 값들은 다음 홀수 행 계산을 위해 내부 메모리에 저장한다.

수평 프로세서는 수직 프로세서로부터 데이터를 받는데, 만약 받아진 입력이 짝수 번째 열의 출력이라면 수평 프로세서는 연산을 수행하지 않고 내부 레지스터에 저장하고, 홀수 번째 열의 출력이면 식 (5)에 나타난바와 같이 행방향으로의 필터링 연산을 수행한다.

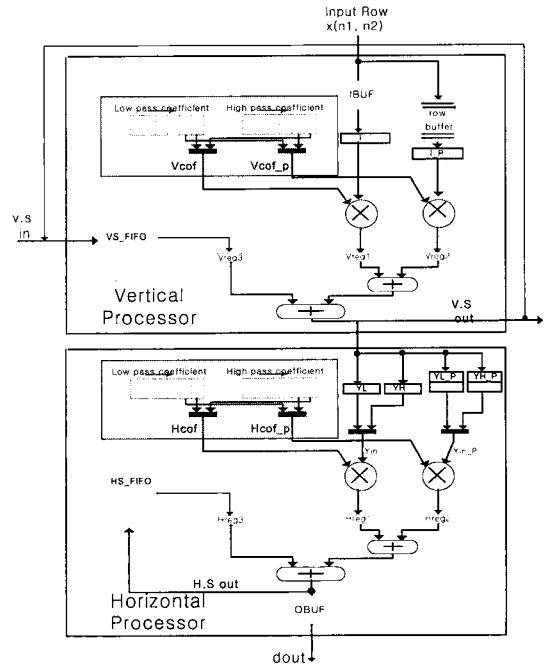


그림 2. 제안된 웨이브렛 변환 프로세서의 블록도
Fig. 2. A block diagram of the proposed Wavelet Transform(WT) processor.

표 1. 웨이브렛 변환 프로세서의 연산 순서
Table 1. Computational sequence of the WT processor.

Cycle	Vcof	Vcof_p	I	I_P	Vreg3	V.S.out	Hcof	Hcof_p	Yin	Y_P	Hreg3	WO
1	H(0)	H(1)	$x(n_1,n_2)$	$x(n_1-1,n_2)$	$q_{2,1}(n_1-2,n_2)^H$	$y_0(n_1,n_2)^H$	-	-	-	-	-	-
2	H(2)	H(3)	$x(n_1,n_2)$	$x(n_1-1,n_2)$	$q_{2,2}(n_1-2,n_2)^H$	$q_{2,1}(n_1,n_2)^H$	H(0)	H(1)	$y_0(n_1,n_2)^H$	$y_0(n_1,n_2-1)^H$	$q_{1,1}(n_1,n_2-2)^{HH}$	$y(n_1,n_2)^{HH}$
3	H(4)	H(5)	$x(n_1,n_2)$	$x(n_1-1,n_2)$	$q_{2,3}(n_1-2,n_2)^H$	$q_{2,2}(n_1,n_2)^H$	H(2)	H(3)	$y_0(n_1,n_2)^H$	$y_0(n_1,n_2-1)^H$	$q_{1,2}(n_1,n_2-2)^{HH}$	$q_{1,1}(n_1,n_2)^{HH}$
4	G(0)	G(1)	$x(n_1,n_2)$	$x(n_1-1,n_2)$	$q_{2,1}(n_1-2,n_2)^G$	$y_0(n_1,n_2)^G$	H(4)	H(5)	$y_0(n_1,n_2)^H$	$y_0(n_1,n_2-1)^H$	$q_{1,3}(n_1,n_2-2)^{HH}$	$q_{1,2}(n_1,n_2)^{HH}$
5	G(2)	G(3)	$x(n_1,n_2)$	$x(n_1-1,n_2)$	$q_{2,2}(n_1-2,n_2)^G$	$q_{2,1}(n_1,n_2)^G$	G(0)	G(1)	$y_0(n_1,n_2)^H$	$y_0(n_1,n_2-1)^H$	$q_{1,1}(n_1,n_2-2)^{GH}$	$y(n_1,n_2)^{GH}$
6	G(4)	G(5)	$x(n_1,n_2)$	$x(n_1-1,n_2)$	$q_{2,3}(n_1-2,n_2)^G$	$q_{2,2}(n_1,n_2)^G$	G(2)	G(3)	$y_0(n_1,n_2)^H$	$y_0(n_1,n_2-1)^H$	$q_{1,2}(n_1,n_2-2)^{GH}$	$q_{1,1}(n_1,n_2)^{GH}$
7	H(0)	H(1)	$x(n_1,n_2+1)$	$x(n_1-1,n_2+1)$	$q_{2,1}(n_1-2,n_2+1)^H$	$y_0(n_1,n_2+1)^H$	G(4)	G(5)	$y_0(n_1,n_2)^H$	$y_0(n_1,n_2-1)^H$	$q_{1,3}(n_1,n_2-2)^{GH}$	$q_{1,2}(n_1,n_2)^{GH}$
8	H(2)	H(3)	$x(n_1,n_2+1)$	$x(n_1-1,n_2+1)$	$q_{2,2}(n_1-2,n_2+1)^H$	$q_{2,1}(n_1,n_2+1)^H$	H(0)	H(1)	$y_0(n_1,n_2)^G$	$y_0(n_1,n_2-1)^G$	$q_{1,1}(n_1,n_2-2)^{HG}$	$y(n_1,n_2)^{HG}$
9	H(4)	H(5)	$x(n_1,n_2+1)$	$x(n_1-1,n_2+1)$	$q_{2,3}(n_1-2,n_2+1)^H$	$q_{2,2}(n_1,n_2+1)^H$	H(2)	H(3)	$y_0(n_1,n_2)^G$	$y_0(n_1,n_2-1)^G$	$q_{1,1}(n_1,n_2-2)^{HG}$	$q_{1,1}(n_1,n_2)^{HG}$
10	G(0)	G(1)	$x(n_1,n_2+1)$	$x(n_1-1,n_2+1)$	$q_{2,1}(n_1-1,n_2+1)^G$	$y_0(n_1,n_2+1)^G$	H(4)	H(5)	$y_0(n_1,n_2)^G$	$y_0(n_1,n_2-1)^G$	$q_{1,1}(n_1,n_2-2)^{HG}$	$q_{1,2}(n_1,n_2)^{HG}$
11	G(2)	G(3)	$x(n_1,n_2+1)$	$x(n_1-1,n_2+1)$	$q_{2,2}(n_1-1,n_2+1)^G$	$q_{2,1}(n_1,n_2+1)^G$	G(0)	G(1)	$y_0(n_1,n_2)^G$	$y_0(n_1,n_2-1)^G$	$q_{1,1}(n_1,n_2-2)^{GG}$	$y(n_1,n_2)^{GG}$
12	G(4)	G(5)	$x(n_1,n_2+1)$	$x(n_1-1,n_2+1)$	$q_{2,3}(n_1-1,n_2+1)^G$	$q_{2,2}(n_1,n_2+1)^G$	G(2)	G(3)	$y_0(n_1,n_2)^G$	$y_0(n_1,n_2-1)^G$	$q_{1,1}(n_1,n_2-2)^{GG}$	$q_{1,1}(n_1,n_2)^{GG}$
13	H(0)	H(1)	$x(n_1,n_2+2)$	$x(n_1-1,n_2+2)$	$q_{2,1}(n_1-2,n_2+2)^H$	$y_0(n_1,n_2+2)^H$	G(4)	G(5	$y_0(n_1,n_2)^G$	$y_0(n_1,n_2-1)^G$	$q_{1,1}(n_1,n_2-2)^{GG}$	$q_{1,2}(n_1,n_2)^{GG}$

<그림 2>는 제안된 웨이브렛 변환 프로세서의 세부 블록도를 나타낸다.

<그림 2>에서 보는 바와 같이 웨이브렛 변환 프로세서 내의 각 프로세서(수직 프로세서, 수평 프로세서)는 식(4), (5)의 연산들을 수행하기 위한 2개의 곱셈기와 2개의 덧셈기를 포함하며, 계산된 상태 변수들을 저장하기위한 내부메모리를 포함한다. N×N 크기의 영상에 대해 수직 프로세서는 짝수 번째 행을 저장하기위한 N크기의 라인메모리와 수직상태 변수들을 저장하기 위한 NK-2N 크기의 FIFO (VS_FIFO) 버퍼를 포함한다. 수평 프로세서는 수평상태변수를 저장하기위한 2K-4크기의 FIFO (HS_FIFO) 버퍼를 포함한다.

<표 1>은 K=6을 가지는 웨이브렛 변환에 대한 프로세서의 연산순서를 보여준다. <표 1>에서 H는 low-pass 필터 계수, G는 high-pass 필터 계수를 나타낸다. 하나의 입력에 사용되는 6연산 싸이클 중 싸이클 1과 싸이클 4에서 수직 프로세서는 입력 $x(n_1,n_2)$에 대한 열방향으로의 필터링 출력(y^H, y^G)을 수평 프로세서로 내보내어 수평프로세서의 YL, YH레지스터에 저장되게 하고, 6싸이클 중 나머지 싸이클에서는 수직 상태변수들을 계산하여 VS_FIFO에 저장한다. 다음 픽셀 $x(n_1,n_2+1)$에 대해서는 싸이클 7과 싸이클 10에서 계산된 필터링 출력을 수평 프로세서의 YL_P, YH_P 레지스터에 저장한다.

수평 프로세서는 수직 프로세서로부터의 열방향 출력이 유용하면 동작할 수 있는데, 싸이클 1에서 계산된 YL내용과 이전 픽셀 $x(n_1,n_2-1)$에 대한 수직프로세서의 low-pass 출력(YL_P레지스터 내용)을 이용하여 싸이클 2와 5에서 행방향으로의 필터링 출력(HH, GH)을 계산한다. 싸이클 8과 11에서는 싸이클 4에서 계산된 YH 레지스터 내용과 이전 픽셀 $x(n_1,n_2-1)$에 대한 수직프로세서의 high-pass 출력(YH_P레지스터 내용)을 이용하여 필터링 출력(HG, GG)을 계산한다. 나머지 싸이클에서는 수평 상태 변수들을 계산한다.

<표 1>에 나타난 바와 같이 제안된 프로세서 구조에서 수직 프로세서는 6 싸이클 마다 한 픽셀씩 처리하고, 수평 프로세서는 2픽셀에 한번 출력하기 때문에 12 싸이클 마다 4개의 출력(HH, GH, HG, GG)을 계산한다. 따라서 제안된 프로세서는 홀 수 행에 대해 6싸이클 마다 한 픽셀을 처리할 수 있고, 짝수 행에 대해서는 연산을 수행하지 않고 수직 프로세서내의 내부메모리에 저장만 하기 때문에, 결과적으로 약 3싸이클 마다 한 픽셀을 처리할 수 있다.

2. 어레이 구성

제안된 프로세서는 필터의 길이 및 영상의 크기에 따라 하나의 프로세서로서도 실시간 처리가 가능하나, 보다 고속 처리가 요구되는 응용 시스템에서는 1차원 어레이 형태의 다중 프로세서로 구성할 수 있다. 이는 블록 데이터 플로우 구조를 따르며, 어레이 구성은 수직프로세서의 수직상태변수 출력(V.Sout)을 인접하는

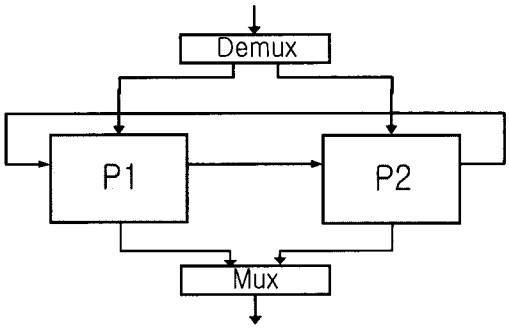


그림 3. 2개의 프로세서로 구성된 어레이 구조
Fig. 3. An array structure consists of two processors.

수직프로세서 내의 FIFO입력(VS_FIFO)에 연결함으로써 구성될 수 있다[8]. <그림 3>은 2개의 프로세서로 구성된 어레이 구조의 한 예를 나타낸다.

어레이 구성시에는 적절한 데이터 분할이 필요하다. 제안된 프로세서 구조에서는 <그림 4>에서와 같이 두 행의 데이터를 하나의 블록 데이터로 취급하며, 수직 상태변수의 데이터 의존성을 고려하여 <그림 4>와 같이 지그 재그(Zig-Zag) 스캔을 수행하면 인접 프로세서(<그림 3>에서 P2 프로세서)의 VS_FIFO 는 한 행에 대한 수직 상태변수 대신에 한 픽셀에 대한 수직 상태 변수만을 저장하면 되므로 인접 프로세서의 VS_FIFO 사이즈는 2K-4 크기를 포함한다.

영상 데이터의 프로세서들로의 할당은 다음과 같다. 첫 번째 블록 데이터는 첫 번째 프로세서에 할당하고, 두 번째 블록 데이터는 두 번째 프로세서에 할당한다. 마지막 프로세서까지 데이터를 할당 하면 다음 블록 데이터는 다시 첫 번째 프로세서에 할당하며 같은 방법으로 계속 수행한다. (warp around 방식).

이는 각 프로세서마다 한 블록 데이터를 처리하기 때문에 프로세서 사이의 데이터 전달 요구가 시스톨릭 어레이에 비해서 훨씬 줄어든다. 더구나 이러한 형태의 데이터 분할 방법은 사용하는 프로세서의 개수가 늘어날수록 비례적으로 성능(throughput)이 향상되므로 매우 효율적인 다중 프로세서 시스템을 구성할 수 있게 해준다. 또한 어레이 구조로 구성할 경우에 개개의 프로세서는 동기적으로 동작하며, 프로세서와 프로세서 간, 프로세서와 입출력 사이는 FIFO 버퍼를 사용함으로써 비동기적으로 동작시킴으로써 동기화 요구를 완곡하게 한다. 전체적으로는 전역 클럭을 사용하지 않으며 프로세서간의 제어가 간단하다.

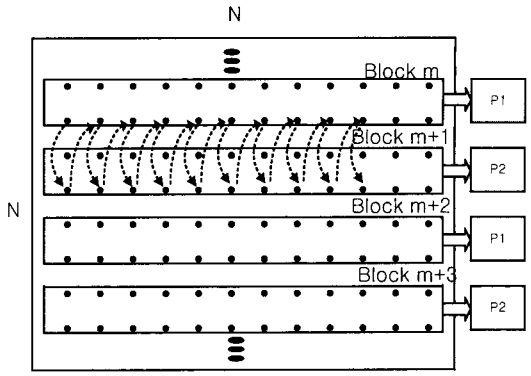


그림 4. 2개의 프로세서로 어레이 구성시의 데이터 분할
Fig. 4. Data decomposition for an array that consists of two processors.

Ⅳ. 검증 및 성능 평가

1. 모델링 및 검증

제안된 웨이브렛 변환 프로세서는 VHDL로 합성 가능한 RTL 수준에서 모델링 되었다. 모델링된 프로세서는 웨이브렛 필터계수로 Daubeach 9-7 필터 계수를 사용하고 256×256영상 크기를 사용한다. 프로세서의 내부 정밀도(precision)는 일반적인 영상 응용에서 주로 사용하는 입력 8비트, 출력 16비트로 결정하였다. 웨이브렛 필터계수의 정밀도는 <그림 5>에 나타난 바와 같이 필터계수가 12비트에서 PSNR 값이 포화(saturation)된다는 점을 고려하여 부호 비트를 포함 12비트로 결정하였다. <그림 5>는 영상처리에서 많이 사용되어지는 Lenna, gold, babara, brige, pepper 등 여러 그레이 영상들에 대해 C 언어에서 수행된 웨이브렛

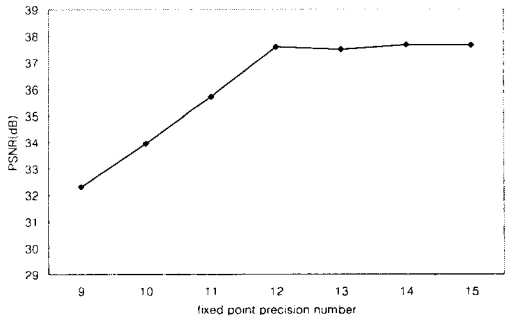


그림 5. 필터 계수의 정밀도에 따른 PSNR
Fig. 5. PSNR due to the precision of filter coefficients.

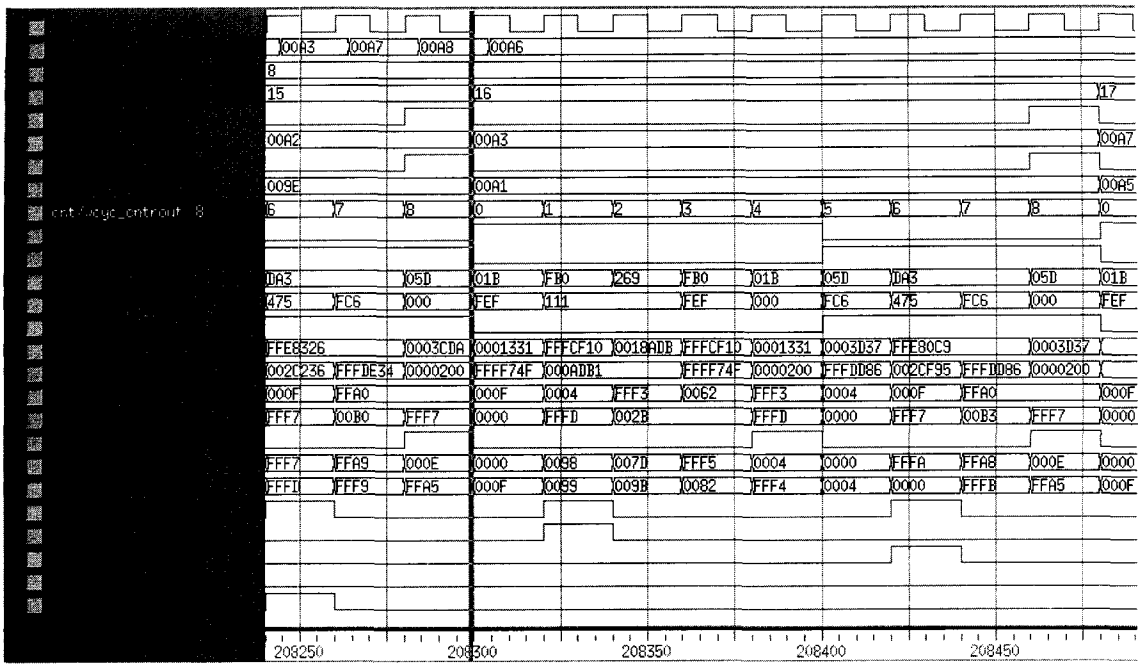


그림 6. 수직프로세서의 시뮬레이션 결과
Fig. 6. Simulation result of the Vertical processor.

필터 계수의 정밀도에 따른 3 분해 레벨에서의 PSNR 값을 나타낸다.

프로세서 내부에 사용되어지는 곱셈기의 경우 성능 및 면적을 고려하여 12×16 수정된 부스 곱셈기를 사용하였으며, 부스 곱셈기 내의 부분곱들을 합하는 트리부에 $512=2^9$를 더하도록 하여 반올림(rounding)을 수행하도록 하였다[9]. 또한 덧셈기는 16비트 CLA를 사용하였다.

모델링된 웨이브렛 변환 프로세서는 대부분의 내장형 시스템에 존재하는 SDRAM과 입출력 인터페이스를 수행하도록 하였는데, SDRAM과의 인터페이스를 위한 SDRAM 제어기는 burst 길이에 따라 2가지 모드를 지원한다. 읽기(Read) 동작시 짝수 행의 경우는 SDRAM에 저장되어진 영상의 한 행을 연속적으로 읽을 수 있는 full page-burst 모드로, 홀수 행에 대해서는 4개의 픽셀을 연속적으로 읽기 위한 4 burst 모드로 읽기 동작을 수행하도록 하였다. 쓰기(Write) 동작시는 홀수행에 대해 프로세서가 동작하는 동안 수행되어지며, 4-burst 모드로 수행하도록 하였다. 이와 같은 SDRAM과의 인터페이스를 고려하여 입력 버퍼로 8×16크기의 FIFO버퍼를 사용하였고, 출력(HH, GH, HG, GG) 버퍼도 각각 8×16크기의 FIFO버퍼를 사용하였다.

<그림 6>과 <그림 7>은 RTL레벨에서 모델링된 프로세서의 Lenna 영상에 대한 VHDL 시뮬레이션 결과들을 나타낸다. <그림 6>은 영상의 9행, 17열(rowcnt = '8', colcnt = '16')의 픽셀에 대한 수직 프로세서의 시뮬레이션 결과를 나타내는데, 영상의입력 값들이 입력 버퍼에 들어오면 수직 프로세서는 수직 싸이클 카운터 (vcyc_cntrout)가 동작하면서 실행하게 된다.

<그림 7>은 수평 프로세서의 시뮬레이션 결과들을 나타내는데, 수평 프로세서는 수직 프로세서에서 YL 값이 나오면(yl_en = '1') 수평 싸이클 카운터(hcyc_cntrout)가 동작하면서 실행하게 된다.

다양한 영상에 대해 검증이 완료된 프로세서는 IDEC에서 제공하는 0.35㎛ CMOS 2-poly 4-metal 현대 팬텀(Phantom) 셀 라이브러리를 사용하여 Synopsys사의 Design Analyzer툴에서 논리 합성되었는데, 수직프로세서내의 행버퍼, 수직 상태변수 메모리는 칩 외부의 SRAM메모리를 사용하도록 인터페이스 하였다. 합성된 프로세서의 게이트 수는 입출력 버퍼들을 포함하여 2 입력 nand 게이트 기준으로 33,019(코어 : 22,261, 입출력버퍼 : 10,758)이고, 동작 주파수는 50 MHz 이다.

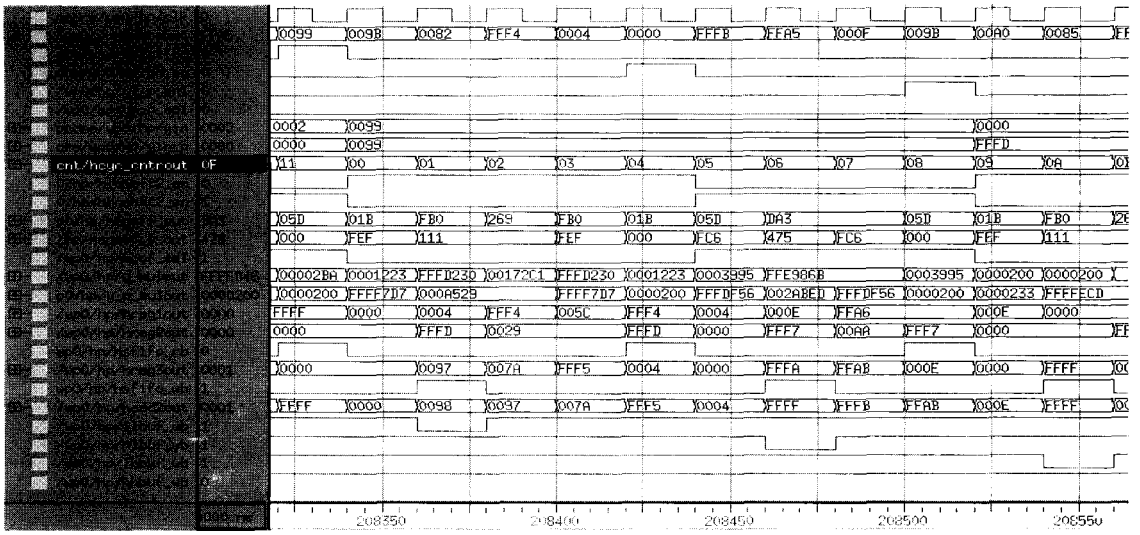


그림 7. 수평프로세서의 시뮬레이션 결과
Fig. 7. Simulation result of the Horizontal processor

그림 7. 수평프로세서의 시뮬레이션 결과
Fig. 7. Simulation result of the Horizontal processor.

2. 성능 분석

2차원 웨이브렛 변환의 응용인 실시간 비디오 처리에서 프로세서는 초당 30 프레임을 처리해야 한다. 이를 위해서는 영상의 한 프레임을 33 ms이하에서 처리해야 하는데, 제안된 프로세서는 필터길이를 K라 할 때 3레벨까지의 모든 상태 변수들과 출력을 계산하는데 $(K/2)[N^2 + N^2/4+N^2/16]$ 싸이클에 처리할 수 있다. 즉 필터 길이가 6이고, 50 MHz 클럭을 사용할 경우 프로세서는 웹캠 등에서 많이 사용되어지는 320×240사이즈의 한 프레임 영상에 대해 6.0 ms에 한 프레임을 처리할 수 있고, RGB 컬러 영상의 경우 18.0 ms가 소요되어 2차원 웨이브렛 변환을 실시간에 처리 할 수 있다. 그러나 512×512 등과 같이 보다 큰 사이즈의 영상의 경우는 20.6 ms가 소요되어, 그레이 영상은 가능하지만 컬러 영상의 경우는 실시간으로 처리할 수 없다. 이때는 2개의 프로세서로 구성된 다중 프로세서를 사용하면 컬러영상의 경우 30.9 ms 가 소요되어 실시간으로 처리할 수 있다.

<표 2>는 제안된 프로세서와 기존에 발표된 구조들과의 성능 비교를 나타낸다. 표에서 알 수 있듯이 다른 구조들은 필터길이에 따라 곱셈기수, 덧셈기수, 필요한 메모리 등의 하드웨어 리소스들이 결정되어지고 수행속도가 고정적인데 반해, 본 구조는 필터길이 뿐만 아니라 프로세서의 수에 따라 하드웨어 리소스 및 수행시간이 결정되어진다. 즉 영상 크기가 대체적으로 적은 내장형 응용분야에 대해 컨볼루션 연산을 병렬로 수행하는 다른 구조들은 수행속도는 빠르나 하드웨어 리소스를 많이 차지하는 반면, 제안된 구조는 영상 크기에 따라 프로세서 수를 정의할 수 있어 하드웨어 리소스를 적게 차지 하면서도 요구되는 성능에 맞출 수 있다.

필요한 메모리에 관련하여 제안된 구조는 프로세서

표 2. 기존 2차원 DWT 구조들과의 성능 비교(M : 프로세서 수)
Table 2. Performance comparison of various architectures for 2-D DWT(M : the number of processor).

	곱셈기수	덧셈기수	필요한 메모리	수행 시간
제안된 구조	4M	4M	$N^2/4+NK-N$	$\approx(K/M)N^2$
병렬-시스톨릭[2]	$K^2/2$	K-1	N(2K-1)	$\approx N^2$
병렬필터[3]	$2K^2$	$2(K^2-1)$	2NK	$\approx N^2$
시스톨릭-병렬[4]	6K	6K	2N(K+2)	$\approx N^2$
시스톨릭 어레이[5]	3K	3K	2NK	$\approx N^2$

내부에 NK-N크기의 메모리 이외에 2번째 분해 레벨 이상을 수행하기위한 LL 밴드 크기인 $N^2/4$ 크기의 외부 메모리를 필요로 한다. 그러나 제안된 구조는 내장형 응용 영상 시스템에 적용되어지고, 이는 기존의 내장형 영상 시스템에 존재하는 SDRAM등의 메모리를 이용할 수 있어 제안된 구조의 실제 시스템에 적용시 따로 $N^2/4$ 크기의 RAM 을 요구하지 않는다.

V. 결 론

본 논문은 상태변수 표현방법을 사용하여 2차원 웨이브렛 변환을 실시간으로 처리 할 수 있는 프로세서 구조를 제안하였다. 제안된 프로세서 구조는 수직프로세서와 수평프로세서로 구성되어지며, 곱셈기, 덧셈기 및 내부 메모리 등의 시스템 리소스를 적게 사용하여 내장형 응용 시스템에 적합하다. 또한 프로세서는 쉽게 다중 프로세서 구조로 확장할 수 있으며, 이때 프로세서들의 수에 따른 선형적인 속도 향상과 함께 고성능을 얻을 수 있어서 다양한 영상 처리 응용 분야에 사용 가능하다.

참 고 문 헌

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

[2] C. Chakrabarti and M. Vishwanath, "Efficient realization of the discrete and continuous wavelet transforms : form single chip implementations to mappings on SIMD array computers", IEEE Trans. Signal Processing, Vol. 43, No. 3, pp. 759-771, Mar. 1995.

[3] Chu Yu and Sao Jie Chen, "VLSI Implementation of 2 D Discrete Wavelet Transform for Real Time Video Signal Processing", IEEE Trans. Consumer Electronics, Vol. 43, No. 4, November 1997.

[4] M. Vishwanath, R. M. Owens, "VLSI Architectures for the Discrete Wavelet Transforms", IEEE Trans. on Circuits and Systems II : Analog and Digital Signal Processing, Vol. 42, No. 5, pp. 305-316, May 1995.

[5] 반성범, 박래홍, 지용, "2차원 이산 웨이브렛 변환을 위한 효율적인 VLSI 구조", 대한 전자공학회 논문지, 제 37권 SP편, 제1호, pp. 59-68, 2000. 1

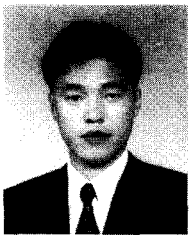
[6] M. Ferretti, D. Rizzo, "A Parallel Architecture for the 2 D Discrete Wavelet Transform with Integer Lifting Scheme", Journal of VLSI Signal Processing, Vol. 28, pp. 165-185, 2001.

[7] S. M. Park, et al., "A novel VLSI architecture for the real time implementation of 2 D signal processing systems", Proc. of Int. Conf. on Computer Design : VLSI in Computers and Processors, pp. 582-585, 1988.

[8] Winser E. Alexander, et al. "Parallel Image Processing with the Block Data Parallel Architecture", Proc. of the IEEE, Vol. 84, No. 7, pp. 947-968, July 1996.

[9] I. S. Abu khater, A. Bellaouar, and M. I. Elmasry, "Circuit Techniques for CMOS Low-Power High Performance Multipliers", IEEE Journal of Solid State Circuits, Vol. 3 1, No. 10, pp. 1535-1546, October 1996.

저 자 소 개



丁 甲 天(正會員)

1996년 : 전남대학교 컴퓨터공학과 학사. 1998년 : 전남대학교 전자공학과 석사. 1999년~현재 : 전남대학교 고품질전기전자부품 및 시스템연구센터 연구원. 1998년~현재 : 전남대학교 전자공학과 박사과정. <주관심분야 : 저전력 프로세서 구조, 영상압축, 영상통신용 ASIC 설계, DSP 설계, VLSI 설계 및 CAD 등임>

朴 性 模(正會員)

1977년 : 서울대학교 전자공학과 학사. 1979년 : 한국과학기술원 전기 및 전자공학과 석사. 1988년 : 노스캐롤라이나 주립대학 전기 및 컴퓨터공학과 공학박사. 1979년~1984년 : 한국전자기술연구소 설계개발부 선임연구원. 1988년~1992년 : 올드도미니언 대학교 전기 및 컴퓨터공학과 조교수. 1992년~현재 : 전남대학교 컴퓨터공학과 교수. 2002년~현재 : 전남대학교 정보전산원 원장. <주관심분야 : 멀티미디어 프로세서 구조, VLSI 시스템 설계, 신호처리용 ASIC 설계, 영상압축, 임베디드 시스템 등>