2003년도 대한전자공학회 하계종합학술대회 논문집 제26권 제1호

5/3필터를 사용한 2차원 DWT에서의 개선된 하드웨어 구조

방정배[0] 정영식 장영조
한국기술교육대학교 정보기술공학부
jbbang@kut.ac.kr sikiga@hanmail.net yjjang@kut.ac.kr

An Improved Hardware Architecture for 2D DWT Using 5/3 Filter

Bang Jeong Bae[0], Jeung Young Sic, Jang Young Jo
Korea University of Technology and Education

요 약

DWT(Discrete Wavelet Transform)를 2차원 하드웨어로 구현하기 위해서 많은 하드웨어와 실행시간이 들기 때문에 효율적인 구조가 중요하다. 그래서, 이 논문에서는 2차원 DWT에 대한 효율적인 하드웨어 이용률과 크기의 감소, 완벽한 레지스터 이용률, 규칙적인 데이터 흐름으로 필터 길이의 확장을 쉽게 할 수 있도록 구조를 개선하고, 개선된 구조를 VHDL로 검증하였다.

I. 서 론

JPEG2000 영상 압축의 중요한 역할을 하는 DWT는 많은 관심을 받아오고 있다. 현재, 2차원 DWT에 대한 VLSI의 구조들은 실시간 처리를 요구하고 있지만, 2차원 DWT는 데이터를 수평과 수직방향으로 필터링 기능을 수행하기 때문에, 구현함에 있어서 계산량이 많아, 빠른 스피드나 저 전력 응용에 낮은 비용으로 높은 효율성을 얻도록 설계한다는 것은 어려운 일이다. 따라서, 이 논문에서 제안된 구조는 이러한 난점을 보완하기 위해 효율적인 하드웨어 이용률, 완벽한 메모리 이용률, 규칙적인 데이터 흐름으로 필터길이의 확장을 쉽게 할 수 있는 구조를 제안하였다.

II. 본 론

2.1 2차원 DWT 알고리즘

그림 1은 2-레벨 2차원 DWT를 설명한 것이다. 각 분해 레벨은 2단계를 거치게 된다. 첫 번째 단계는 행으로 필터링을 하고, 두 번째 단계는 열로 필터링을 하게 된다. 영상의 크기가 N×N이면, 첫 번째 레벨 분해에서 N/2×N/2의 크기를 갖는 LL ,LH, HL, HH 네 개의 부밴드가 출력이 되고, 두 번째 레벨 분해에서 LL인 데이터를 입력 받아 N/4×N/4의 크기를 갖는 LLLL, LLLH, LLHL, LLHH 네 개의 부 밴드가 출력이 된다. 이런 방법으로 입력 데이터 IN[m][n]에 대한 수학적 수식으로 표현 한다면 다음과 같다.

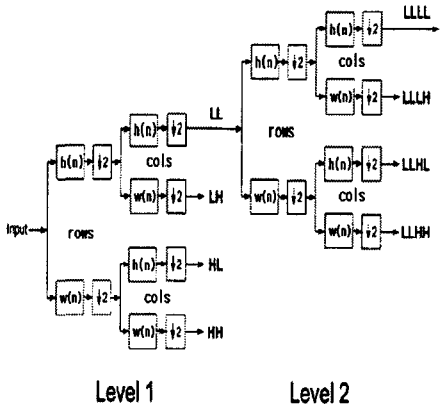

그림 1. 2-레벨 2차원 DWT

$$L_{j+1}[row][m] = \sum_{i=0}^{N_L-1} w[i] \times LL_j[row][2m-1]$$

$$H_{j+1}[row][m] = \sum_{i=0}^{N_H-1} h[i] \times LL_j[row][2m-1-i]$$

$$LL_{j+1}[n][col] = \sum_{i=0}^{N_L-1} w[i] \times L_j[2n-1][col]$$

$$LH_{j+1}[n][col] = \sum_{i=0}^{N_H-1} h[i] \times L_j[2n-1-i][col]$$

$$HL_{j+1}[n][col] = \sum_{i=0}^{N_L-1} w[i] \times H_j[2n-1][col]$$

$$HH_{j+1}[n][col] = \sum_{i=0}^{N_H-1} h[i] \times H_j[2n-1-i][col]$$

여기서,

$j = [0, 1, \ldots, L-1]$, $row = [0, 1, \ldots, N/2^j - 1]$

$m = [0, 1, \ldots, M/2^{j+1} - 1]$

$col = [0, 1, \ldots, N/2^j - 1]$,

$n = [0, 1, \ldots, N/2^{j+1} - 1]$,

$LL_0[n][m] = IN[n][m]$

2.2 병렬 필터 구조

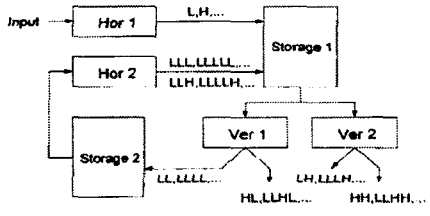

그림 2. 병렬 필터 구조

현재, 2차원 DWT에 대한 널리 알려진 구조가 그림 2에 있는 병렬 필터 구조 이다. 이 구조는 1차원 DWT 구조에서 초기에 제안된 구조인 MRPA(Modified Recursive Pyramid Algorithm)을 근거로 한 것이다. 위 구조는 4개의 필터와 2개의 메모리로 구성되어있다. Hor 1은 첫 번째 레벨에서만 행으로 필터링 하고, Hor 2는 두 번째 레벨에서 행으로 필터링하며 다음 레벨들을 실행하게 된다. Ver 1과 Ver 2는 모든 레벨에서 열로 필터링을 실행한다. 1 레벨 2차원 DWT에서 병렬 필터 구조의 하드웨어 이용률은 단지 절반에 불과하고, 레벨이 증가함에 따라 하드웨어 이용률이 비효율적이기 때문에 병렬 필터 구조뿐 만 아니라 현재 2차원 DWT 구조의 주요 문제점인 많은 계산 시간을 요구한다.

2.3 2차원 DWT 구조

2.3.1 개선 전 전체 구조(5/3 필터)

위 구조는 참고문헌 [1]의 구조를 5/3필터로 구현하였을 경우를 표현한 구조이다. stage1에서는 짝수 주기 동안과 홀수 주기 동안 입력 데이터와 계수 값들의 곱과 합으로 데이터의 값을 출력하게 되고, stage2에서는 한 쌍의 곱셈기와 덧셈기, 레지스터를 공유하며 사용하고, 스위치로 데이터의 흐름을 제어함으로써 데이터 값을 출력하게 된다. 위 구조를 살펴보게 되면 필터의 계수가 늘어날 경우에 stage 2의 레지스터 수는 (N/2)X4 만큼 늘어나게 되고, 레지스터 이용률이 전체 66.7%로 낮아지게 되는 단점이 있기 때문에, 앞으로 참고문헌 [1]의 구조를 가지고 5/3필터에 적합한 구조로 개선하여 레지스터 이용률과 레지스터 수, 하드웨어 크기를 줄일 목적으로 Stage2 부분을 중점적으로 다루었다.

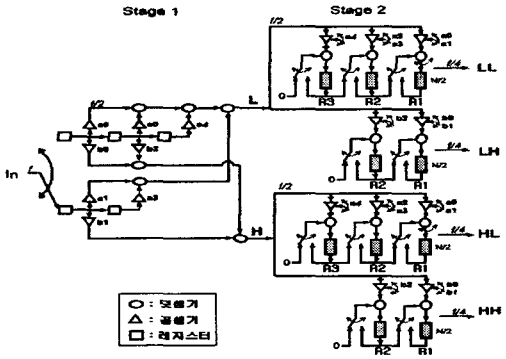

그림 3. 5/3 필터 Transform Module

2.3.1.1 개선 전 Lowpass 부분

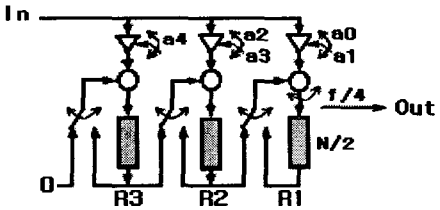

그림 4. Lowpass 부분

그림 4는 lowpass부분만 표현한 것이며, 아래(표1)는 데이터 흐름을 나타낸 것이다.

표 1. 데이터 흐름

Clk	SW	In	R3	R2	R1	OUT
0	0	$x(0)$		$a_3x(0)$	$a_1x(0)$	
1N	1	$x(1)$	$a_4x(1)$	$a_2x(1)+a_3x(0)$	$a_0x(1)+a_1x(0)$	
2N	0	$x(2)$		$a_3x(2)+a_4x(1)$	$a_1x(2)$	
3N	1	$x(3)$	$a_4x(3)$	$a_2x(3)+a_3x(2)+a_4x(1)$	$a_0x(3)+a_1x(2)+a_2x(1)+a_3x(0)$	
4N	0	$x(4)$	$a_3x(4)$	$a_3x(4)+a_4x(3)$	$a_1x(4)+a_2x(3)+a_3x(2)a_4x(1)$	
5N	1	$x(5)$	$a_4x(5)$	$a_2x(5)+a_3x(4)+a_4x(3)$	$a_0x(5)+a_1x(4)+a_2x(3)+a_3x(2)+a_4x(1)$	
6N	0	$x(6)$		$a_3x(6)+a_4x(5)$	$a_1x(6)+a_2x(5)+a_3x(4)+a_4x(3)$	
7N	1	$x(7)$	$a_4x(7)$	$a_2x(7)+a_3x(6)+a_4x(5)$	$a_0x(7)+a_1x(6)+a_2x(5)+a_3x(4)+a_4x(3)$	

(N : 열 데이터 길이)

데이터는(2i+1)N(i=0,1,2...N/2-1) 클럭 주기마다 출력이 된다.

2.3.1.2 개선 전 Highpass 부분

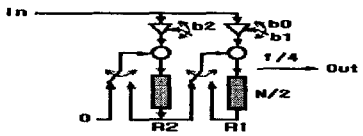

그림 5. Highpass 부분

그림 5는 highpass부분만 표현한 것이며, 아래(표2)는 데이터 흐름을 나타낸 것이다.

표 2. 데이터 흐름

Clk	SW	In	R2	R1	OUT
0	0	$x(0)$		$b_1 x(0)$	
1N	1	$x(1)$	$b_2 x(1)$	$b_0 x(1)+ b_1 x(0)$	
2N	0	$x(2)$		$b_1 x(2)+ b_2 x(1)$	
3N	1	$x(3)$	$b_2 x(3)$	$b_0 x(3)+ b_1 x(1)$	
4N	0	$x(4)$		$b_1 x(4)+ b_2 x(3)$	
5N	1	$x(5)$	$b_2 x(5)$	$b_0 x(5)+ b_1 x(4)+ b_2 x(3)$	
6N	0	$x(6)$		$b_1 x(6)+ b_2 x(5)$	
7N	1	$x(7)$	$b_2 x(7)$	$b_0 x(7)+ b_1 x(6)+ b_2 x(5)$	

데이터는(2i+1)N(i=0,1,2...N/2-1) 클럭 주기마다 출력이 된다.

2.3.2 개선된 Lowpass 부분

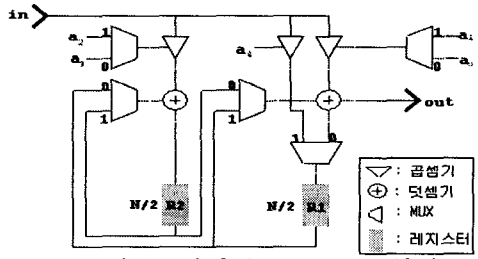

그림 6. 개선된 Lowpass 부분

그림6에서 볼 수 있듯이 개선 전과 비교해 볼 때, 레지스터 수를 N/2만큼 줄였으며, 아래(표3)에서 보는 바와 같이 100%의 레지스터 이용률을 볼 수 있다

개선 전 구조를 보게 되면 R1에서 데이터 출력할 때 생기는 빈 공간을 MUX를 사용하여 R3에 저장하게 될 데이터를 R1에 저장하고, 저장한 데이터를 와이어로 R2에 연결함으로써 데이터 충돌이나 빈 공간 없이 이용률을 향상 시키면서 레지스터 수 또한 줄일 수 있게 된다.

표 3. 데이터 흐름

clk	sw	In	R2	R1	out
0	0	$x(0)$	$a_3 x(0)$	$a_1 x(0)$	
1N	1	$x(1)$	$a_2 x(1)+ a_3 x(0)$	$a_4 x(1)$	$a_0 x(1)+ a_1 x(0)$
2N	0	$x(2)$	$a_3 x(2)+ a_4 x(1)$	$a_1 x(2)+ a_2 x(1)$ $+ a_3 x(0)$	
3N	1	$x(3)$	$a_2 x(3)+ a_3 x(2)$ $+ a_4 x(1)$	$a_4 x(3)$	$a_0 x(3)+ a_1 x(2)$ $+ a_2 x(1)+ a_3 x(0)$
4N	0	$x(4)$	$a_3 x(4)+ a_4 x(3)$	$a_1 x(4)+ a_2 x(3)$ $+ a_3 x(2)+ a_4 x(1)$	
5N	1	$x(5)$	$a_2 x(5)+ a_3 x(4)$ $+ a_4 x(3)$	$a_4 x(5)$	$a_0 x(5)+ a_1 x(4)+ a_2 x(3)$ $+ a_3 x(2)+ a_4 x(1)$
6N	0	$x(6)$	$a_3 x(6)+ a_4 x((5$ $)$	$a_1 x(6)+ a_2 x(5)$ $+ a_3 x(4)+ a_4 x(3)$	
7N	1	$x(7)$	$a_2 x(7)+ a_3 x(6)$ $+ a_4 x(5)$	$a_4 x(7)$	$a_0 x(7)+ a_1 x(6)+ a_2 x(5)$ $+ a_3 x(4)+ a_4 x(3)$

2.3.3 개선된 Highpass 부분

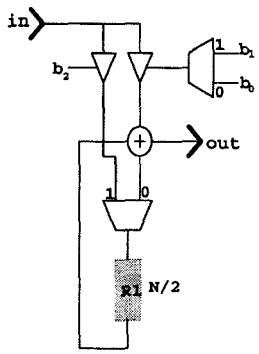

그림 7. 개선된 Highpass 부분

Lowpass 부분과 같은 방식으로 레지스터 수를 N/2만큼 줄였으며, 아래(표4)에서 보는 바와 같이 100%의 레지스터 이용률을 볼 수 있다

표 4. 데이터 흐름

clk	sw	In	R1	out
0	0	$x(0)$	$b_1 x(0)$	
1N	1	$x(1)$	$b_2 x(1)$	$b_0 x(1)+ b_1 x(0)$
2N	0	$x(2)$	$b_1 x(2)+ b_2 x(1)$	
3N	1	$x(3)$	$b_2 x(3)$	$b_0 x(3)+ b_1 x(2)+ b_2 x(1)$
4N	0	$x(4)$	$b_1 x(4)+ b_2 x(3)$	
5N	1	$x(5)$	$b_2 x(5)$	$b_0 x(5)+ b_1 x(4)+ b_2 x(3)$
6N	0	$x(6)$	$b_1 x(6)+ b_2 x(5)$	
7N	1	$x(7)$	$b_2 x(7)$	$b_0 x(7)+ b_1 x(6)+ b_2 x(5)$

Lowpass 부분과 같이 R1에서 데이터 출력할 때 생기는 빈 공간을 MUX를 사용하여 R2에 저장하게 될 데이터를 R1에 저장하고, 저장한 데이터를 와이어로 덧셈기에 연결함으로써 데이터 충돌이나 빈 공간 없이 이용률을 향상 시키면서 레지스터 수 또한 줄일 수 있게 된다.

2.3.4 Stage2 부분의 성능 비교

표 5 . 성능 비교

	개선 전	개선 후	개선률
레지스터 수	N/2X10	N/2X6	40%
클럭개수	10	10	0%
덧셈기 수	10	6	40%
MUX수	20	14	30%
평 균	-	-	27.5%

표 5에서 보듯이 레지스터 수, 덧셈기 수, MUX의 수를 비교해 볼 때, 개선 후가 3가지 면에서 더 향상된 것을 볼 수 있다.

2.3.5 VHDL로 구현한 전체 블록도

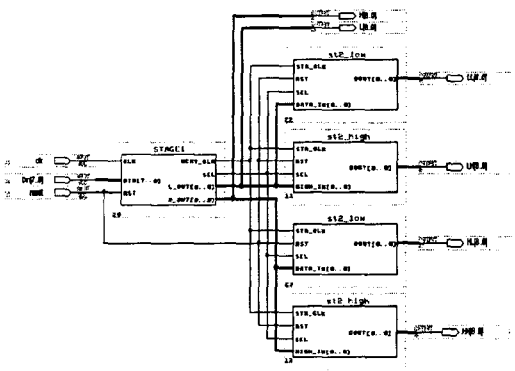

그림 8. VHDL로 구현한 전체 블록도

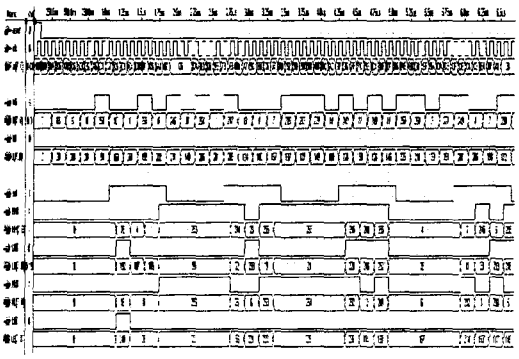

그림 9. 결과 시뮬레이션

위 그림은 개선된 구조를 가지고 VHDL로 전체블록도와 그에 대한 결과 시뮬레이션을 나타낸 그림이다. 시뮬레이션은 8X8 크기의 영상 값을 입력으로 12클럭에 4개의 부 밴드 값을 첫 출력으로 시작하여 sel 신호가 High일 때 출력하게 된다. 출력 값은 계산 값과 거의 일치하는 값으로 시뮬레이션 파형이 나타났다.

III. 결 론

이 논문에서는 2차원 DWT에 대한 효율적인 하드웨어 이용률, 완벽한 레지스터 이용률과 평균 27.5% 크기의 감소율, 규칙적인 데이터 흐름으로 필터길이의 확장을 쉽게 할 수 있도록 구조를 개선하였고. 개선된 구조를 VHDL로 검증하였다. 필터의 계수가 짝수개면 그만큼 레지스터 이용률이 낮아지지만, 5/3필터나 9/7필터를 사용할 경우 하드웨어 이용률 면에서 효율적이고, 메모리 이용률 면에서는 100%가 되기 때문에 앞으로 사용함에 있어서 더욱 효율적이다.

참고 문헌

[1] Po-Cheng Wu and Liang-Gee Chen. Fellow. IEEE, " An Efficient Architecture for Two-Dimensional Discrete WaveletTransform", IEEE Trans, Circuits Syst. Video Technol. vol 11, pp. 536-545, April 2001.

[2] N.D. Zervas, G.P. Anagnostopoulous, V.Spiliotopoulos, Y.Andreopoulos, and C.E.Goutis, "Evaluation of Design Alternatives for the 2D Discrete Wavelet Transform", IEEE Trans. Circuits Syst. Video Technol., vol 11, pp. 1246-1262, Dec. 2001.

[3] C.Chakrabarti and C.Mumford, "Efficient Realization of Analysis and Systhesis Filters Based on The 2-D Discrete Wavelet Transform". IEEE International Conferencen, vol 6, pp. 3256-3259, May 1996.

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