JPEG2000 을 위한 Lifting-based Forward DWT 의 VLSI 구조

김정욱, 강형주, 박인철
한국과학기술원 전자전산학과 전기및전자공학부
전화 : (042) 869-4405

A VLSI Architecture of Lifting-based Forward DWT for JPEG2000

Jung-Wook Kim, Hyeong-Ju Kang, In-Cheol Park
Department of Electronics and Electrical Engineering, KAIST
E-mail : {jwkim,dk,icpark}@ics.kaist.ac.kr

Abstract

This paper proposes a new VLSI architecture suitable for the lifting-based forward DWT of JPEG2000. The architecture consists of a column-processor and a row-processor, and requires a smaller buffer between column and row than the previous works. The output of the proposed DWT can be directly fed to the input of EBCOT block that is at the next of DWT in JPEG2000 standard. The architecture was designed and synthesized in Verilog HDL. The estimated area and operating frequency in 0-25um technology are 13071gates and 220MHz, respectively.

I. 서론

Discrete wavelet transform(DWT)는 90 년대 후반 이후로 영상의 부호화에서 수요가 끊임없이 증가하여 최근에는 정지영상 압축기술인 JPEG2000 으로 표준으로 지정되었다. JPEG2000 은 기존의 정지영상 압축기술인 JPEG 보다 더 높은 압축비와 동일한 압축비에서 월등히 높은 화질을 보여준다.

그러나 DWT 는 JPEG2000 encoder 의 첫 번째 블록으로 기존 영상압축 방식에서 사용되었던 DCT 와 비교하여 더 높은 연산량과 메모리를 요구한다. 따라서 디지털 카메라와 같은 휴대용 기기에서는 JPEG2000 의 신속한 encoding 과 decoding 를 수행하기 위해 DWT 를 위한 전용 hardware 가 요구된다.

기존의 DWT 는 영상을 Low-pass filter(LPF)와 High-pass filter(HPF)의 두 개의 filter bank 를 각각 통과시킨 후 down-sampling 을 수행하며, LPF 의 결과 영상은 다시 이 과정을 반복한다. Filter 는 Finite Impulse Response(FIR) filter 의 형태로 구성이 되어 있다. 기존의 DWT 방식은 FIR 의 수식을 그대로 구현한 Convolution-based DWT 가 많았으나[1][2] 95 년에 소개된 Lifting-based DWT 방식[3][4]은 기존의 Convolution-based DWT 방식에 비해 연산량과 메모리 bandwidth 를 크게 줄이는데 공헌하였다.

영상처리를 위한 2D DWT 는 column 이나 row 방향으로 1D DWT 를 수행한 후, 반드시 직각 방향으로 1D DWT 를 수행해야 한다. 따라서 여러 2D DWT architecture 에서는 row 방향으로 1D DWT 연산을 수행한 후 buffer 에 저장하고, column 방향으로 1D DWT 연산을 수행하는 방식을 선택하고 있기 때문에 row 방향의 1D DWT 연산기와 column 방향 1D DWT 연산기 사이에 큰 크기의 buffer 를 요구했다. 이 논문에서는 row 방향 1D DWT 와 Column 방향 1D DWT 의 관계를 이용하여, 적은 양의 buffer 를 사용하는 2D DWT 의 구조를 제안한다.

이후 논문은 2 장에서 lifting-based DWT 알고리즘에 대한 간단한 소개를 한다. 3 장에서는 제안된 DWT 연산기의 Architecture 를 설명한다. 4 장에서는 제안된 DWT 연산기를 Verilog HDL 로 구현, 기존의 방식과 비교하고, 5 장에서 결론에 대해서 논한다.

II. Lifting-based DWT

DWT 는 하나의 high-pass filter 와 하나의 low-pass filter 로 구성되어 있다. Lifting-based scheme 은 HPF 와 LPF 로 구성된 DWT 의 matrix 를 factorizing 하여 upper matrix 와 low matrix, diagonal matrix 의 조합으로 표현함으로써 연산의 양을 줄이게 된다.

$h(z)$ 를 HPF, $g(z)$ 를 LPF 라 하면 DWT 의 기본적인 연산 matrix 는 다음과 같이 표현된다.

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

위와 같은 연산은 DWT filter 의 특성인 biorthogonality 를 이용하여[4], 다음과 같은 연산으로 분리할 수 있다.

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

Factorizing 한 matrix 를 이용하여 Filtering 은 Predict, Update, Scaling 의 step 을 가지게 되며, 각 step 에서는 연산을 위해서 자신과 주변의 두 sample 만이 필요하다.

1) Predict step $y_{2n+1}=x_{2n+1}+\alpha(x_{2n}+x_{2n+2})$
2) Update step $y_{2n}=x_{2n}+\beta(y_{2n-1}+y_{2n+1})$
3) Scaling step $z_{2n}=Ky_{2n}$, $z_{2n+1}=\frac{1}{K}y_{2n+1}$

Lifting-based DWT 에서는 predict stage 에서는 odd sample 을 계산하고, update stage 에서는 even sample 을 계산하여, 이 두 sample 간의 연관성을 이용함으로써, 연산량을 줄이게 된다.

표 1 은 JPEG2000 의 표준으로 채택된 DWT filter 인 (5,3)와 (9,7)의 두 방식에서 연산량 비교를 보여주고 있다.[5]

또한, Lifting-based DWT는 forward DWT를 수행하는 hardware구조에서 덧셈을 뺄셈으로만 전환하면, inverse DWT를 수행할 수 있으므로, FDWT/IDWT를 동시에 집적할 경우 hardware cost를 추가적으로 줄일 수 있다.

Filter	Multiplication		Addition	
	Convolution	Lifting	Convolution	Lifting
(5,3)	4	2	6	4
(9,7)	9	5	14	8

표 1. Convolution-based 방식과 Lifting-based 방식의 연산량 비교

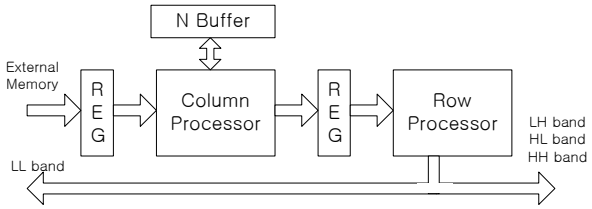

그림 1. 제안된 Architecture의 Block Diagram

III. 제안된 Architecture

A. Memory Access

제안된 architecture는 Memory에서 N x N image를 가져와서 3개의 별도의 memory bank에 각각 LH band와 HL band와 HH band의 계수를 저장한다. 저장된 계수들은 JPEG2000의 encoder에서 DWT의 다음 block인 EBCOT encoder가 읽어서 최종적인 JPEG2000 bit-stream을 형성하게 된다. EBCOT encoder는 subband 2개에서 계수를 받아서 연산을 수행하지 않으므로, Forward DWT의 결과는 각각의 bank에 저장되는 것이 좋다.

LL band의 계수는 역시 별도의 memory bank에 저장되었다가, 더 높은 level의 DWT를 수행하기 위해서 다시 읽혀지고, 이 DWT 결과는 마찬가지로 band 별로 다른 bank에 저장되게 된다.

B. Processing Sequence and Buffer Size Decision

제안된 DWT 구조는 그림 1과 같은 구조를 가지고 있다. 이 구조는 크게 Column-Processor(CP)와 Row-Processor(RP)로 구성되어 있으며, 그 사이에 register를 삽입하여 pipeline으로 Column에서 연산한 결과를 곧바로 Row에서 처리한다. 다른 여러 구조와 달리 Column의 처리를 먼저 하게 된 것은 JPEG2000 encoder에서 DWT와 EBCOT encoder사이의 buffer의 size와 latency를 줄이기 위해서이다.

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63	67
68	72	76	80	84												
69	73	77														

그림 2. 16x16 Coding Block의 EBCOT coding 순서.

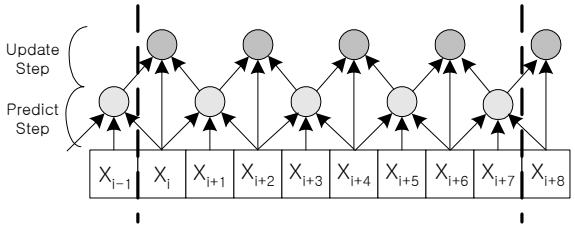

그림 3. Lifting-based DWT에서 5/3 filter의 data dependency

그림 2에서 보듯이 JPEG2000의 EBCOT에서는 세로 방향으로 4줄씩 읽고, 가로 방향으로 한 칸 이동하는 방식으로 encoding 순서가 정해져 있다. 기존의 DWT 구조에서는 Row부터 연산결과가 나오므로, 최소 N x N 의 image의 경우 최소 3N 의 DWT 계수를 저장할 수 있는 memory가 필요하다. 그러나 coding block의 순서와 일치하여 DWT 계수가 나온다면, image 크기와 관계없이 8개의 buffer만이 존재하면 된다.

각 subband에서 4개의 세로방향 계수가 필요하므로, CP에서는 8개의 sample을 처리하고 가로방향으로 이동한다. 이 때, 계수 중 가장 아래 계수는 CP가 다시 같은 Column으로 돌아왔을 경우, 연산을 위해 필요하므로 buffer에 저장해 두었다가 다시 불러서 사용한다.

Buffer의 크기는 CP가 읽어야 하는 Memory의 Bandwidth의 크기와 밀접한 관련이 있다. 그림 3에서 보듯이 5/3 filter에서는 경계에 있는 sample x_i 의 DWT 계수를 구하기 위해서는 x_{i-2} 부터 x_{i+2} 가 필요하다. 그러나 x_{i-2} 과 x_{i-1} 는 CP가 읽었던 sample이다. 이를 다시 memory에서 읽지 않기 위해서는 x_{i-1} 의 Predict step 까지의 결과를 저장하는 buffer를 Row의 크기만큼 삽입하면 된다. 만약 buffer를 사용하지 않는다면, x_{i-2} 와 x_{i-1} 를 다시 읽어야 하므로, buffer는 사라지게 된다.

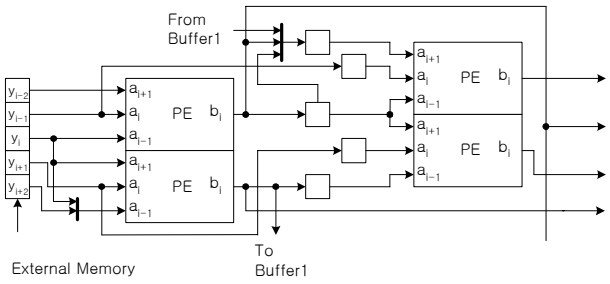

그림 4. Column-Processor의 구조

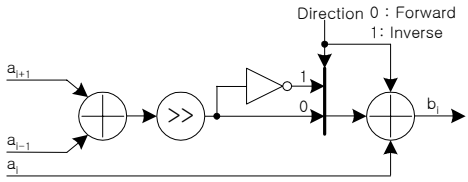

그림 5. Lifting-based DWT에서의 Processing element (5/3 filter)

5/3 filter의 경우는 buffer를 없앴을 경우, 두 번 읽게 되는 sample의 숫자가 한 CP의 연산 당 3 samples 이지만, 9/7 filter의 경우는 7 samples 가 되기 때문에, original image size의 두 배에 가까운 memory를 access를 필요로 하게 된다. 따라서, buffer의 크기와 CP의 memory bandwidth간의 적절한 trade-off가 필요하다.

C. Pipeline

CP와 RP는 각각 Predict stage와 Update stage간에 다시 pipeline 구조를 이루고 있다. 그리고 CP의 앞과 RP의 뒤에 Memory Access를 위한 pipeline stage를 각각 추가하여 6 단계의 pipeline stage를 가지고 있다.

CP와 RP는 JPEG2000의 표준인 5/3 filter와 9/7 filter 양쪽을 모두 지원하기 위해서 4개의 processing element를 가지고 있다. 9/7 filter의 경우는 predict1 → update1 → predict2 → updage2 → scaling 의 순서를 거치기 때문에 각각의 processing element가 predict와 update step 등을 맡게 된다.

IV. 구현 및 비교

A. Precision Analysis

제안된 architecture에서 JPEG2000에서 lossless DWT filter로 사용되는 5/3 filter를 구현하였다. Filter에서 곱해

지는 상수는 소수이므로, fixed point에서 정확한 연산을 위해서는 LSB에 bit를 추가하여 왼쪽으로 shift해야 한다. 그러나 너무 많이 shift할 경우는 DWT의 면적증가와 속도 저하에 영향을 끼치게 되므로 적절한 bit를 LSB에 추가해야 한다.

표 2는 C++을 이용한 Simulation 결과이다. RTL 구현시의 additional bit는 정확도를 위해서 2bit로 정하였다.

	Additional Bit			
Image	0	1	2	3
Lena 512	37.91	47.95	Inf	Inf
Lena 256	38.03	48.23	Inf	Inf
barbara 512	37.81	47.91	Inf	Inf
barboon 256	37.97	48.04	Inf	Inf
Bridge 256	37.98	48.11	Inf	Inf
PEPPER 256	38.24	48.22	Inf	Inf

표 2. 여러 test 영상에 대해서 5-level 5/3 filter를 통해 FDWT/IDWT를 수행했을 때의 PSNR (단위 : dB)

B. 구현 결과

제안한 architecture에서 5/3 filter를 verilog HDL로 구현하여 동작을 검증하였다. 이 회로를 삼성 0.25um 공정에서 합성한 결과 220MHz의 동작 주파수에서 내부 메모리와 Buffer를 제외한 core의 면적이 13071gates로 나타났다.

표 3은 Column Processor와 Row Processor에서 processing element가 2개씩만 사용될 경우, 제안한 Architecture를 다른 architecture와 비교한 결과이다.

	K.Andra[5]	C-J. Lian[6]	K-C. B. Tan[7]	Proposed
Adders	8	8	5	8
Shifters	4	4	5	4
Buffer	4N	N x N	3N	N
RPMEM	$N \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$

표 3. 기존의 architecture와의 비교

RPMEM은 LL band를 recursive하게 DWT하기 위한 memory이다. 제안된 architecture에서 buffer의 size가 가장 작으며 이것은 memory의 bandwidth와 교환이 가능하다.

V. 결론

DWT는 JPEG2000의 표준에서 사용되는 연산으로 많은 연산량과 메모리를 필요로 한다. 우리는 이것을 해결하기 위해서 기존의 architecture에서 존재하였던 column 방향과 row 방향의 연산을 이어주는 buffer를 없애거나 줄일 수 있는 architecture를 고안하였고, buffer의 size는 유동적이 될 수 있음을 확인하였다. 또한 JPEG2000의 표준에 적합한 DWT 연산 순서를 통해 JPEG2000 encoding system에서 적은 양의 buffer와 세밀한 pipeline을 구현할 수 있게 하였다.

Reference

[1] M. Vishwanath, R. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform", IEEE Trans. on Circuits and Systems: II, May 1995

[2] K.K. Parhi and T.Nishitani, "VLSI architectures for discrete wavelet transforms", IEEE Transaction on VLSI Systems, Vol. 1, pp.191-202, June 1993

[3] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions", Proceedings of SPIE, Vol. 2569, 1995, pp68-79

[4] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting schemes," The Journal of Fourier Analysis and Applications, Vol. 4, pp.247-269, 1998

[5] K. Andra, C. Chakrabarti, and T. Acharya, “A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform,” IEEE Trans. on Signal Processing, April 2002

[6] C-J. Lian, K-F Chen, H-H Chen and L-G Chen, “Analysis and architecture design of lifting based DWT and EBCOT for JPEG2000”, Proc. of Technical Papers for 2001 International Symposium on VLSI Technology, Systems and Applications, pp. 180-183, 2001

[7] K-C. B. Tan and T.Arslan, “Shift-Accumulator ALU Centric JPEG2000 5/3 Lifting-based Discrete Wavelet Transform Architecture”, IEEE International Conference on Circuits and Systems 2003