논문 01-26-8B-11 한국통신학회논문지 '01-8 Vol.26 No.8B

실시간 이차원 웨이블릿 변환의 FPGA 구현을 위한 효율적인 메모리 사상

준회원 김 왕 현*, 서 영 호*, 김 종 현*, 정회원 김 동 욱*

The Effcient Memory Mapping of FPGA Implementation for Real-Time 2-D Discrete Wavelet Transform

Wang-Hyun Kim*, Young-Ho Seo*, Jong-Hyun Kim* *Associate Members*

Dong-Wook kim* *Regular Member*

요 약

본 논문에서는 이차원(2-D) 이산 웨이블릿 변환(Discrete Wavelet Transform, DWT)을 이용한 연상압축기를 FPGA 칩에서 실시간으로 동작 가능하도록 하는 효율적인 메모리 스케줄링 방법(E^2M^2)을 제안하였다. S/W적으로 위의 메모리 사상 방법을 검증한 후, 실제로 상용화된 SDRAM을 선정하여 메모리 제어기를 구현하였다. 본 논문에서는 Mallet-tree를 이용한 2-D DWT 영상압축 칩을 구현할 경우를 가정하였다. 이 알고리즘은 연산 과정에서 많은 데이터를 정장하여야 하는데, FPGA는 많은 데이터를 저장할수 있는 메모리가 내장되어 있지 않으므로 외부 메모리를 사용하여야 한다. 외부메모리는 열(row)에 대해서만 연속(burst) 읽기, 쓰기 동작이 가능하기 때문에 Mallet-tree 알고리즘의 데이터 입출력을 그대로 적용할 경우 실시간 동작을 수행하는 DWT 압축 칩을 구현할 수 없다. 본 논문에서는 데이터 쓰기를 수행할 경우에는 메모리 셀(cell)의 수직 방향을 저장시키고 읽기를 수행할 때는 수평으로 데이터의 연속 읽기를 수행함으로써 필터가 항상 수평 방향에 위치하게 하는 방법을 제안하였다. 입 방법을 C-언어로 DWT 커넬(Kernel)과 메모리의 에뮬레이터(emulator)를 구현하여 실험한 결과, Mallat-tree 이론을 그대로 적용시켰을 때와 동일한 필터링을 수행할 수 있음을 검증하였다. 또한, 상용화된 SDRAM의 메모리 제어기를 H/W로 구현하여 시뮬레이션 함으로서 본 논문에서 제안한 방법이 실제적인 하드웨어로 실시간 동작을 할 수 있음을 보였다.

ABSTRACT

This paper proposed an efficient memory scheduling method(E^2M^2), with which the real-time image compression using 2-dimensional discrete wavelet transform(2-D DWT) is possible in an FPGA chip. After proving the memory mapping method in software, the memory controller was designed targeting an commercial SDRAM IC. In this paper, we assumed that the 2-D DWT was performed as the Mallat-tree. Because this algorithm needs to store much data during the computation process but FPGA does not have enough memory in general, some external memories have to be used. As an external, memory can perform the burst read or write only for row data, it is not possible to implement a real-time image DWT compressor if the memory read/write of the Mallat-tree algorithm is applied. So, this paper prposed the method to write data into memory cells in vertical direction during DWT and to read out by busrt-read operation so that the filter is always applied in horizontal direction. The experimental result from applying the proposed method to the DWT kernel and memory emulator

* 광운대학교 공과대학 반도체 및 신소재 공학과

논문번호 : K01125-0501, 접수일자 : 2001년 5월 1일

※ 이 논문은 2001년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.

implemented in C-language proved that the proposed scheme produces the same result to the case applying the Mallat-tree algorithm in software. Also, the designed memory controller for an commercial memory IC was implemented and simulated to show that the proposed scheme in this paper could be implemented and operated in real-time.

I. 서 론

최근 멀티미디어 정보에 대한 욕구가 증가하면서 영상/비디오 신호 처리의 속도문제와 고압축률로 영상을 압축할 때의 영상 화질개선에 대한 많은 연구가 진행되고 있다. 현재의 영상/비디오 처리를 위한 신호처리 방법은 DCT(Discrete Cosine Transform)를 기반으로 하는 방법들이 주로 사용되 있는데(JPEG, MPEG등)[1][2], 8×8 화소(pixel) 단위로 처리되는 DCT는 압축률이 증가함에 따라 블록효과(blocking artifacts)가 나타나는 문제가 발생한다. 최근 웨이블릿(wavelet)을 이용한 이산 변환방법(Discrete Wavelet Transform, DWT)이 대두되어 그 사용 영역을 넓혀가고 있다[1][2][5][6]. DWT는 음성,영상, 비디오 등 다방면에서 효과적인 신호 압축 방법으로 연구되고 있으며, 이미 JPEG2000에 표준 변환방법으로 채택되어 있다[7]. 웨이블릿 변환은 기저함수의 길이가 가변적이므로 DCT와는 달리 블록 효과를 제거할 수 있다는 장점 외에 부대역(sub-band)별 처리(압축률 조정 가능 등)가 가능하고 고압축률로 영상을 압축할 때 DCT보다 우수한 성능을 보인다[8][9][10].

그러나, 2차원(two-dimension, 2-D) DWT는 연산 과정상에서 많은 데이터를 저장해야 한다. 따라서, 실제 하드웨어(hardware, H/W) 구현에 있어서 메모리와 연산 모듈사이에 데이터를 주고받는 시간이 전체 H/W의 성능을 크게 좌우하게 된다.[8] H/W는 사용용도와 칩의 생산량에 따라 ASIC 혹은 FPGA 등으로 구분된다. 종래에는 대량생산을 목적으로 할 때 설계기간과 비용의 부담은 커지나 ASIC으로 하드웨어를 구현하였으나, 근래에는 FPGA 칩 가격이 많이 하락하여 H/W 설계 시 처음부터 FPGA를 타겟으로 구현하고 있는 추세에 있다[11][12]. 그러나, 2-D DWT와 같이 연산과정에서 데이터를 많이 저장하고 불러 사용하여야 하는 알고리즘을 FPGA로 구현할 경우, 내부 메모리의 부족으로 외부 메모리를 사용해야 되는 문제점이 발생한다. 외부 메모리로는 구현의 용이성 및 FPGA 특성으로 인해 대체로 SDRAM을 사용한다. SDRAM은 상용화되어 있는 것을 사용하므로 메모리 접근시간(access time)이 제한되어 있다. 따라서, 제한된 접근시간을 최대한 이용하여 데이터를 효율적으로 사상(mapping)하는 방법이 전체 하드웨어의 실시간 동작에 영향을 주는 중요한 부분이 된다. 상용화되어 있는 SDRAM은 제어신호와 동작모드를 세팅하는 방법이 유사하므로 본 논문에서 제안하는 메모리 사상 방법은 FPGA상에서 2-D DWT를 구현할 때 어떠한 상용 메모리에서도 적용할 수 있으리라 사료된다.

본 논문에서는 2-D DWT를 이용하여 영상을 압축할 때, H/W 성능에 가장 큰 영향을 미치는 메모리 사상 방법에 대한 처리 문제를 다룬다. 기존에 웨이블릿과 관계된 알고리즘[13]~[15]들은 데이터 처리의 성능 향상과 중간의 데이터를 적게 저장하면서 연산을 수행할 수 있는 방법에 주안점을 두었을 뿐 실제 H/W 구현시 발생하는 메모리 사상방법에 대해서는 크게 다루지 않았다. 또한, 근래에 Mallat 알고리즘을 수정한 RPA[16]와 line을 기반으로 해서 열(row)의 연산을 수행하면서 출력되는 웨이블릿 계수의 수가 행(column)을 연산할 정도가 되면 행에 대한 변환을 동시에 수행하여 메모리 사용량을 줄이는 알고리즘[8][17]이 제안되었으나, 실제 H/W로의 구현에 주안점을 두지 않고 S/W적인 데이터 읽기/쓰기 동작 횟수에만 관심을 두었다. 본 논문에서는 H/W 구현의 용이성에 초점을 두어 원래의 Mallat 알고리즘을 하드웨어로 구현하였을 때를 가정한다.

실제 FPGA 구현 시 사용하는 SDRAM은 열 방향에 대해서만 연속(burst)읽기와 쓰기가 가능하도록 설계되어 있다. 2-D DWT의 연산 특성상 연산 과정에서 행 방향의 데이터를 연속적으로 읽어오지 못함으로서 데이터의 실시간 처리에 문제를 발생시킨다. 즉, 이론상으로 데이터를 메모리에 사상하는 방법과 실제 하드웨어 구현에 있어서 메모리 사상 방법이 다르다는 것이다. 본 논문에서 제시한 방법(E^2M^2, Efficient External Memory Mapping)은 데이터 쓰기 과정을 열에 따라서 사상하지 않고, 다음 연산을 고려하여 하나의 열에 쓰고 그 다음 데이터는 그 다음 열에 쓰는 과정을 수행함으로써, 다음 연산을 수행할 때 데이터 읽기를 하나의 열에 대해 연속 읽기를 수행하여 연산하는 모듈이 실시간 처리를 할 수 있게 해준다.

본 논문에서 제안한 E^2M^2방법은 Daubechies의 (9,7)필터를 사용한 2-D DWT에 적용하며 NTSC방식의 640×480 영상을 웨이블릿 알고리즘으로 압축할 때 외부 메모리를 사용하는 경우 C언어를 사용하여 만든 메모리 에뮬레이터(emulator)를 사용하여 메모리 읽기/쓰기 동작의 데이터 사상을 검증한다. 또한 H/W적으로 100MHz의 클럭(clock)으로 동작하는 상용 512K × 16bit × 2Banks SDRAM을 제어하는 메모리 제어기를 VHDL 행위-레벨로 구현하여 시뮬레이션을 수행함으로써 실제 FPGA상에서의 동작을 검증한다. 본 논문의 메모리 제어방식은 영상압축을 실시간 처리한다는 가정 하에 수행되며, 이를 위한 메모리 읽기/쓰기 횟수를 계산하고 이에 따라 메모리 제어기를 설계한다.

Ⅱ. Mallat tree를 이용한 2-D DWT 알고리즘

본 논문에서는 Mallat-tree형식의 DWT를 가정하며,Daubechies(9,7) bi-orthogonal 필터를 사용하는 것으로 한다. Mallat-tree DWT 알고리즘은 주어진 영상을 주파수 성분에 따라 연차적으로 분할하여 영상의 에너지를 저주파 영역으로 집중시키는 작용을 한다. 그 수행 방법은 식 (1)과 (2)에 나타낸 것과 같이 영상의 고주파성분과 저주파성분을 각각 필터링(filtering)한다[18].

$$y_{low} = \sum_{n=0}^{N-1} x(2k-n)h(n) \quad (1)$$

$$y_{high} = \sum_{n=0}^{N-1} x(2k-n)g(n) \quad (2)$$

이 수식에 의해서 한 개의 화소가 DWT되는 방법은 그림 1과 같다. 이 과정은 (9,7)필터를 사용할 경우 각 화소를 필터링하기 위해서 좌우로 이웃한 4개의 픽셀들이 필요하므로 총 9개의 화소를 필터 계수와 곱한 후 그 값을 더하는 과정으로 이루어진다. 2-D DWT에서는 연산 과정을 열과 행에 대해서 수행하게 되는데, 이 연산을 수행하는 H/W의 블록도를 그림 2에 나타내었다. 이러한 하드웨어의 구조를 MAC(Multiplicator-and-Accumulator)이라고 부르며, 화소의 값과 필터계수를 곱하는 곱셈기와 곱한 결과의 축적덧셈을 수행하는 덧셈기, 그리고 축적덧셈의 값을 보유하는 레지스터로 이루어져 있다.

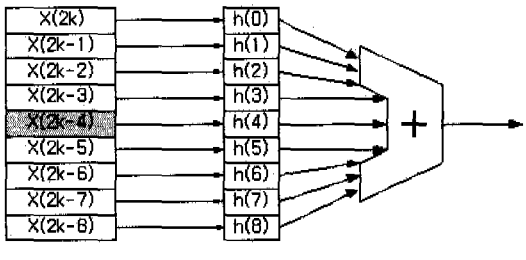


그림 1. (9,7) 필터 탭에 의한 영상의 필터링

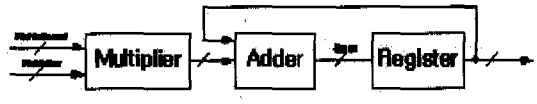


그림 2. MAC의 하드웨어 구조

인간 눈의 특성에 따라 저주파 영역에 시각적으로 중요한 정보가 집중되므로 Mallat-tree 알고리즘은 저주파 영역에 대한 신호 분할 및 서브-샘플링(sub-sampling)을 계속 수행한다. 그림 3은 Mallat-tree를 이용하여 영상의 열을 기준으로 필터링을 수행하고 그 결과를 행 방향으로 필터링할 때, 네 레벨까지 2-D DWT를 수행하는 절차와 그 결과 분할된 영상을 보여주고 있다[6]. 4-레벨 수행 결과는 총 13개의 부대역으로 나뉘어지며, 각 부대역은 전체 영상에 대한 특별한 영상변화 정보들을 갖게 된다. 이 때, 행 연산이 수행되기 위해서는 열에 대해 변환이 수행된 웨이블릿 계수들을 메모리에 저장한 후, 이 값을 읽어서 행 방향 필터링을 수행하게 된다. 각 레벨의 열과 행 필터링을 수행한 결과는 총 4개의 영역(LL, LH, HL, HH)으로 분할되며, 이 중에서 LL영역이 그 다음 레벨의 변환에 사용된다. 따라서, 메모리를 읽고 쓰는 과정이 매우 많이 필요하며, H/W로 DWT를 구현하는 경우 연산을 수행

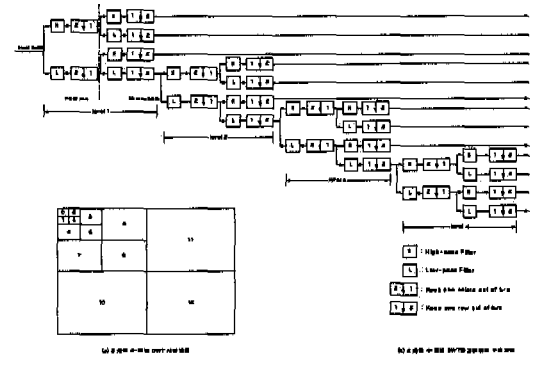


그림 3. 2차원 DWT의 4-레벨 수행절차와 영상분할 결과

하는 MAC(그림 2)의 동작속도와 함께 메모리의 읽기/쓰기 시간은 H/W의 성능과 직접 관련되며, H/W 복잡도 증가에도 큰 영향을 미친다.

Ⅲ. 메모리 읽기/쓰기 시간

본 논문에서는 DWT를 수행할 대상영상을 현재 가장 널리 사용되는 NTSC 방송영상의 크기를 가정하며, Y:Cb:Cr=4:2:2인 컬러정보로 구성되었다고 가정한다. 일반적으로 N×M(N: 행의 개수, M: 열의 개수)의 컬러 영상에 대해 DWT를 수행할 때 한 개의 MAC이 사용된다면 MAC이 처리하여야 하는 Y(DY_{total}), Cb(DCb_{total}), Cr(DCr_{total}) 각각과 전체 데이터 양($DAll_{total}$)은 식 (3), (4), (5), (6)과 같다.

$$DY_{total}=\frac{(N\cdot M)}{2}\cdot 2+\frac{(N\cdot M)}{8}\cdot 2 + \frac{(N\cdot M)}{32}\cdot 2+\frac{(N\cdot M)}{128}\cdot 2=\frac{85}{64}(N\cdot M) \quad (3)$$

$$DCb_{total}=\frac{(N\cdot M)}{4}\cdot 2+\frac{(N\cdot M)}{16}\cdot 2 + \frac{(N\cdot M)}{64}\cdot 2+\frac{(N\cdot M)}{256}\cdot 2=\frac{85}{128}(N\cdot M) \quad (4)$$

$$DCr_{total}=\frac{(N\cdot M)}{4}\cdot 2+\frac{(N\cdot M)}{16}\cdot 2 + \frac{(N\cdot M)}{64}\cdot 2+\frac{(N\cdot M)}{256}\cdot 2=\frac{85}{128}(N\cdot M) \quad (5)$$

$$DAll_{total}=DY_{total}+DCb_{total}+DCr_{total} =\frac{340}{128}(N\cdot M)\simeq 2.66(N\cdot M) \quad (6)$$

예를 들어, NTSC 방식의 640×480 영상에 대해 한 개의 MAC으로 처리하여야 하는 데이터 량과 MAC의 동작 속도는 다음과 같다. 위의 일반식에서 MAC의 데이터 처리량을 구해보면 2.66×(640×480) ≃816,000이다. 따라서 1초에 30프레임(frame)의 컬러영상을 처리하기 위해서 MAC 1개가 한 화소를 처리하여야 하는 시간은,

$$\frac{1}{30\times 816000}\simeq 41[ns]$$

이다. 이 시간은 MAC이 9번의 곱셈과 8번의 덧셈을 수행하여야 하는 시간으로 실제 칩에서 이러한 성능을 가진 MAC을 구현하기는 불가능하다. 따라서 실제 H/W 구현에 있어서는 여러 개의 MAC이 사용되며, MAC 각각에 대한 데이터 처리 순서를 결정하여 DWT를 수행하게 된다. 일반적으로 Y성분의 경우 고역통과(high-pass) 필터링과 저역통과(low-pass) 필터링을 모두 수행하나, 실험결과 Cb와 Cr의 경우에는 인간의 눈이 컬러정보에 민감하지 않기 때문에 저역통과 필터링만을 수행한다. 그러므로, 본 논문에서는 Y성분에 2개, Cb 성분에 1개, Cr성분에 1개, 총 4개의 MAC을 사용하는 것으로 한다. 이 경우 한 화소의 처리 시간은 Y와 Cb/Cr 모두 약 163[ns]가 필요하게 된다. 즉, MAC 1개당 163ns의 시간내에 1개의 화소를 변환하면 된다.

또한, MAC이 필터링을 수행하기 위해서는 전 단계에서 수행한 결과를 메모리로부터 읽어 들이고, 현재 단계에서 수행한 결과를 메모리에 저장하여야 한다. 이 때, 640×480 각 레벨에서 MAC 한 개당 메모리 접근수를 구해 보면 다음과 같다.

Y성분에 대한 MAC의 메모리 접근수(MY_{total})는 다음과 같다.

Level 1 : 640×480(쓰기) + 640×480 (읽기) + 320×240(쓰기)

Level 2 : 320×240(읽기) + 320×240 (쓰기) + 320×240(읽기) + 160×120(쓰기)

Level 3 : 160×120(읽기) + 160×120 (쓰기) + 160×120(읽기) + 80×60(쓰기)

Level 4 : 80×60(읽기) + 80×60(쓰기)+ 80×60 (읽기) + 40×30(쓰기)

MY_{total} = 691,200 + 249,600 + 62,400 + 15,600 = 1,018,800

Cb 및 Cr성분에 대한 MAC의 메모리 접근수(MCb_{total}, MCr_{total})은 다음과 같다.

Level 1 : 320×480(쓰기) + 320×480 (읽기) + 160×240(쓰기)

Level 2 : 160×240(읽기) + 160×240 (쓰기) + 160×240(읽기) + 80×120(쓰기)

Level 3 : 80×120(읽기) + 80×120 (쓰기) + 80×120(읽기) + 40×60(쓰기)

Level 4 : 40×60(읽기) + 40×60 (쓰기) + 40×60(읽기) + 20×30(쓰기)

MCb_{total} 또는 MCr_{total} = (345,600 + 124,800 + 31,200 + 7,800) = 509,400

$MAll_{total}$ = MY_{total} + MCb_{total} +MCr_{total}= 2,037,600

위의 계산 과정을 살펴보면 각 레벨에서 변환될 때 메모리의 쓰기와 읽기 과정이 동시에 발생한다는 것을 알 수 있다. 그러나 현재 영상처리 분야에서 가장 많이 사용되는 상용 메모리는 입출력 데이터 버스가 하나인 것이다. 이 때, 한 개의 메모리로 쓰기와 읽기 동작을 모두 수행할 경우 위에서 계산한 163[ns] 연산시간은 필요한 데이터를 읽어오고 처리한 데이터를 쓰기를 수행하기에는 턱없이 부족하다. 따라서 여러 개의 메모리를 사용하여 특정 레벨에서 각 메모리는 쓰기 동작 또는 읽기 동작만을 수행하도록 하여야 한다.

또한 Y, Cb, Cr은 모든 레벨에서 반드시 동시에 처리되어야 하는 것은 아니다. 예를 들어 아날로그 카메라에 의해 포획된 영상은 디지털 신호로 변환된 후 그 디지털 데이터를 사용하여 영상처리를 하게 된다. 이 때 현재 가장 많이 사용되는 A/D변환기의 출력신호는 Y:Cb:Cr=4:2:2인경우 Cb-Y-Cr-Y-Cb-Y-Cr-Y…로 구성된다. 따라서 첫 번째 레벨에서는 Y, Cb, Cr을 동시에 처리할 수밖에 없고 그 다음 레벨부터는 Y, Cb, Cr을 각각 따로따로 처리하는 것이 효과적이다. 본 연구에서는 그림 4에 나타낸 것과 같이 데이터를 처리하는 순서를 정의하여 처리하였다. 이와 같이 MAC의 데이터 처리 순서가 결정되고 사용되는 메모리 개수를 결정하면 메모리 사상맵을 만들 수 있다. 본 논문에서는 실시간 처리

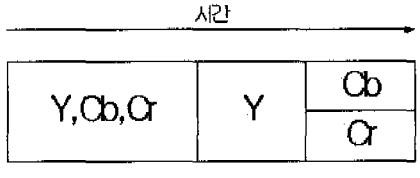


그림 4. Y.Cb,Cr에 대한 MAC들의 데이터 처리 순서

표 1. 4-MAC,4-메모리의 경우 메모리 사상법

Level	row/col	Memory 1	Memory 2	Memory 3
Y,Cb,Cr level 1	row	Write-Y	Write-Cb,Cr	store next frame
	col	Read-Y	Write-Y	
Y-level 2	row	Write-Y	Read-Y	
	col	Read-Y	Write-Y	
Y-level 3	row	Write-Y	Read-Y	
	col	Read-Y	Write-Y	
Y-level 4	row	Write-Y	Read-Y	
	col	Read-Y	Write-Y	
Cb,Cr level 1	col	Write-Cb,Cr	Read-Cb,Cr	
Cb,Cr level 2	row	Read-Cb,Cr	Write-Cb,Cr	
	col	Write-Cb,Cr	Read-Cb,Cr	
Cb,Cr level 3	row	Read-Cb,Cr	Write-Cb,Cr	
	col	Write-Cb,Cr	Read-Cb,Cr	
Cb,Cr level 4	row	Read-Cb,Cr	Write-Cb,Cr	
	col	Write-Cb,Cr	Read-Cb,Cr	

를 위해 4개의 메모리를 사용하는 것으로 하였으며 이 때의 메모리 사상맵은 표 1과 같다. 표 1에서 메모리 1과 메모리 3은 교대로 다음 프레임을 저장하여 FPGA칩이 30프레임/초의 속도로 동영상을 실시간 처리할 수 있도록 해준다.

Ⅳ. 메모리 사상(mapping)방법

상용 SDRAM은 그림 5와 같은 동작 상태에 따라 메모리 제어 신호들과 데이터를 메모리에 입력함으로서 데이터 읽기나 쓰기를 수행한다. 그림 5의 'command' 상태에서 메모리 주소(address)가 입력되는데, 이 때 하나의 주소를 입력하면 그 주소로부터 2개, 4개, 8개 또는 한 페이지(page)를 연속적으로 읽거나 쓰는 동작을 수행하는 기능이 있는데 이를 연속동작(burst operation)이라 한다. 일반적으로 상용 SDRAM에서는 열 방향에 대해서만 이 기능을 제공한다.

이론상으로 열 방향의 변환을 수행할 때는 필터가 수평으로 대응되게 되고, 행 방향으로 처리할 때는 필터가 수직 방향으로 놓이게 된다. 따라서, 열(행) 방향 변환한 결과를 그대로 저장한다면 그 다음 단계의 행(열) 방향 변환을 수행할 때 필요한 데이터는 그림 6에서 보인 바와 같이 사용방향과 반대의 방향에 놓이게 된다. 따라서 필요한 수만큼의 데이터를 한 개씩 읽어와야 하므로 DWT를 수행하는 속도에 큰 영향을 주게된다. 특히 상용 메모리 SDRAM을 사용할 경우에는 그림 5에 나타낸 것과 같이 매번 메모리 읽기 또는 쓰기를 수행할 때마다 row-active, command, precharge 동작을 수행하므로 매우 큰 메모리 접근시간이 요구된다.

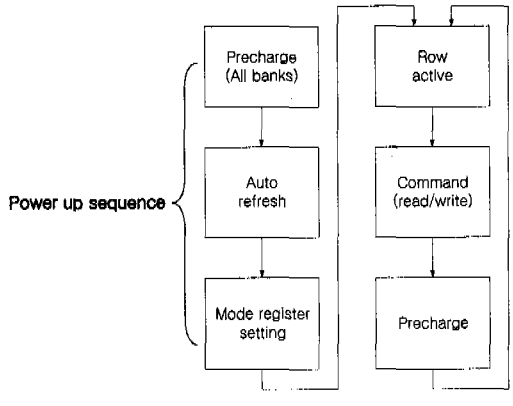


그림 5. SDRAM의 동작 과정

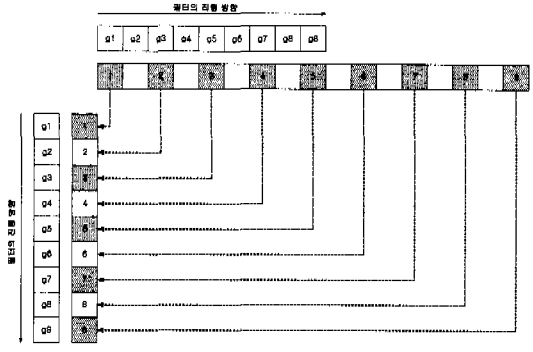


그림 6. 이론적인 웨이블릿 계수의 메모리사상 방법

본 논문에서는 이 문제를 해결하기 위해서 4개의 MAC으로부터 출력되는 웨이블릿 계수 값을 메모리에 저장할 때, 다음 과정의 처리와의 상관관계를 고려하여 2개씩 쌍을 만들어 그림 7과 같이 2개의 열에 연속 2로 저장한다.

즉, 행은 고정시키고 열을 1씩 증가시키면서 연속 2로 쓰고, 열을 다 채우면 행주소를 2 증가시켜 위의 과정을 반복하는 방식이다. 이 방법으로 데이터 쓰기를 수행하면 2개의 열에 있는 데이터가 하나의 그룹이 된다. 예를 들어, 그룹 1의 MAC 1 데이터는 이론적으로 데이터를 수직으로 사상했을 때와 같은 상관관계를 갖게 된다. 따라서, 데이터를 읽어올 때는 그림 8과 같이 상용 메모리의 특성을 이용하여 열 방향으로 연속 8로 읽을 수 있게 되어 163ns안에 9개의 데이터를 MAC에 입력시킬 수 있다. 또한 데이터를 읽어올 때 한 개의 그룹에서 4번 스캔(scan)한 후에 다음 그룹으로 이동하여야 한다.

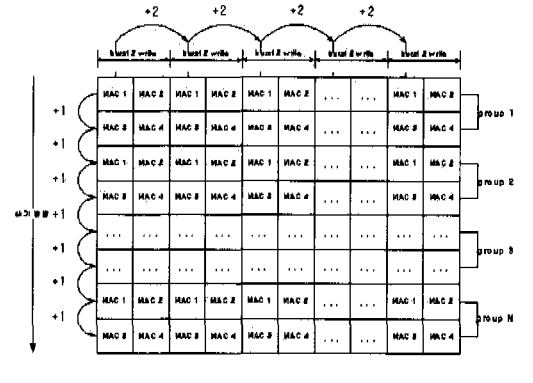


그림 7. 본 논문에서 제안한 메모리 쓰기방법

왜냐하면, 그림 9에서 보다시피 한 개의 그룹을 1차원으로 펼쳤을 때가 이론적인 필터링을 수행할 때 1개의 열에 해당하기 때문이다. 그림 9와 같은 형식으로 데이터가 MAC에 들어가기 위해서는 MAC과 메모리 사이에 다중화기(multiplexer)가 있어서 연속 8로 읽어 온 데이터를 분할하여 버퍼에 입력시키는 모듈이 필요하다.

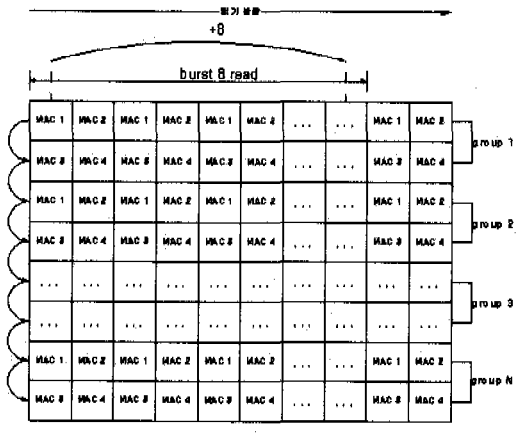


그림 8. 본 논문에서 제안한 메모리 읽기방법

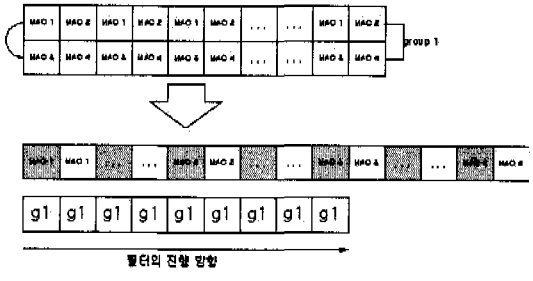


그림 9. 1차원으로 데이터를 배열한 경우

V. S/W 및 H/W 구현 및 결과

이상에서 설명한 본 논문의 메모리 사상방법을 구현함에 있어서 본 논문의 내용이 영상압축과 밀접한 관계를 가지므로 영상압축률에 직접적인 영향을 미치는 양자화(quantization) 과정을 포함하는 처리 과정을 구현하였는데, 그림 10에 이 과정을 개략적으로 나타내었다. 본 논문에서는 그림 중 음영으로 표시한 부분만을 구현하였다. 일반적으로는 양자화 과정 뒤에 엔트로피 코딩을 수행하는 것이 보통이며, 이 과정에서 추가적인 압축효율을 얻을 수 있다. 그러나, 본 논문에서 의도하는 메모리 사상방법과는 관련이 없으므로 제외시켰다. 따라서, 본 논문에서 구현한 부분은 그림 10중 음영부분이다.

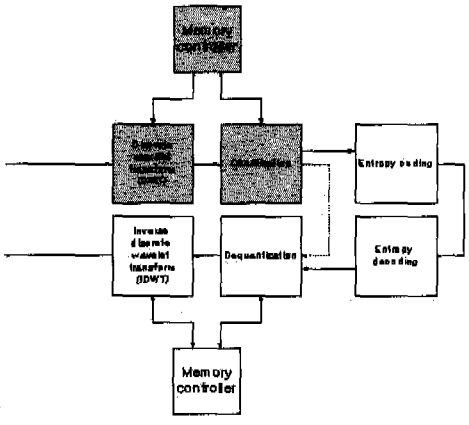


그림 10. 본 논문의 영상처리 과정

양자화기는 H/W 구현의 용이성을 고려하여 그림 11에 나타낸 고정 스칼라 양자화기(fixed linear quantizer)를 설계하여 구현하였다. 이 양자화기는 다수의 영상을 대상으로 최적의 양자화 범위와 압축률에 대한 각 부대역의 할당 비트수를 구하여 설계하였다. 이러한 수행과정에 대해 S/W적으로는 C-언어를 사용하여 구현하였으며, H/W적으로는 VHDL 행위-레벨로 설계한 후 합성하는 방법을 사용하였다.

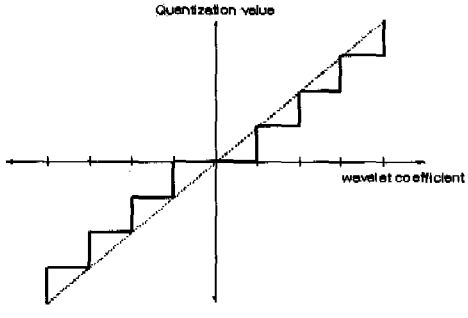


그림 11. 설계된 고정 스칼라 양자화기

H/W 구현에 있어서 대상 플랫폼은 Altera FPGA를 선택하였으며, 설계 툴로는 MAX Plus II를 사용하였고 외부 메모리로는 삼성 512K × 16bit × 2banks SDRAM을 사용하였다. 구현에 사용된 필터로는 Daubechies의 (9,7) bi-orthogonal 필터를 사용하여 NTSC 형식의 640×480 영상을 4-레벨까지 DWT를 수행하는 것으로 하였다.

먼저, C-언어를 이용하여 구현한 MAC, 버퍼 및 다중화기와 메모리 에뮬레이터를 사용하여 본 논문

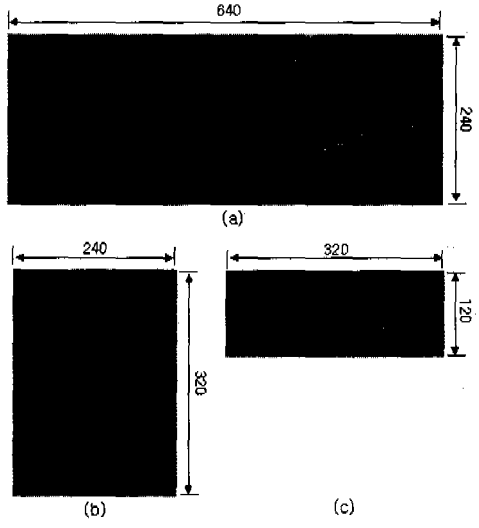


그림 12. C- 언어 에뮬레이터에 의한 레벨1의 실험결과
(a) 원 영상의 짝수 필드
(b) 레벨 1의 열방향 필터링 결과
(c) 레벨 1의 행방향 필터링 결과

에서 제시한 메모리 사상방법을 적용, 레벨 1의 열과 행의 DWT를 수행한 결과를 그림 12에 나타내었다. 열 방향의 DWT를 수행할 때 MAC에서 나온 결과를 $+90^0$만큼 회전시켜 연속 2로 저장하였고, 그 다음 과정인 행 방향의 DWT를 수행할 때는 MAC에서 나온 결과를 다시 -90^0만큼 회전시켜 저장하여야 다음의 열 방향 DWT를 수행할 수 있게 하였다. 결과에서 보다시피 본 논문에 서 제안한 방법이 기능적으로 DWT 알고리즘 연산에 아무런 영향을 주지 않음을 알 수 있다.

그림 13에는 본 논문에서 제시한 방법으로 데이터를 메모리에 읽기 또는 쓰기할 수 있게 해주는 메모리 제어기를 H/W로 설계한 결과를 나타내었다.

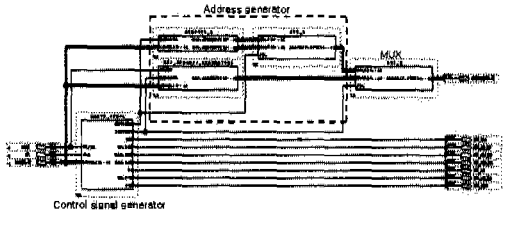


그림 13. 설계된 메모리 제어기의 Altera MAX Plus+II 합성결과

이 H/W는 크게 address generator와 control signal generator의 두 부분으로 나뉜다. Address generator는 각 레벨에서 DWT에 필요한 주소를 생

성하고, control signal generator는 각 메모리의 동작상태에 따라 메모리에 직접적인 제어신호를 만들어 공급한다. 그림 13의 H/W는 Altera FLEX10KE EPF10K100EFC256-1에서 약 12% 즉, 692셀을 사용했다. 100Mhz의 주파수로 동작하여야 하는 부분은 control signal generator이며, 타이밍 분석을 수행한 결과 125Mhz의 클럭 주파수에서 동작하였다. 그리고, 주소 발생기의 경우에는 control signal generator에서 생성하는 클럭에 의해서 동작하기 때문에 100Mhz로 동작할 필요가 없다. 타겟 SDRAM의 클럭 주파수(100Mhz) 와 시퀀스간의 time을 고려했을 때, 레벨 1에서 write를 할 때의 시뮬레이션 결과를 그림 14에 나타내었다. invoke2 신호에 의해서 모드 레지스터는 연속동작의 종류를 선택하게 되고, invoke1에 의해서 일반적인 명령(wirte,read)에서 필요한 주소가 발생하게 되며, 이 외의 출력 신호들은 메모리를 제어하는 제어신호이다.

그림 14에서 볼 수 있듯이, 150ns내에 쓰기 2번을 연속 2로 수행 할 수 있으므로, 앞서 III장에서 계산한 MAC의 동작 속도를 만족시킨다. 따라서, 본 논문의 메모리 사상방법을 이용한 메모리 제어기는 실시간 웨이블릿 FPGA칩에 적용될 수 있음을 알 수 있다.

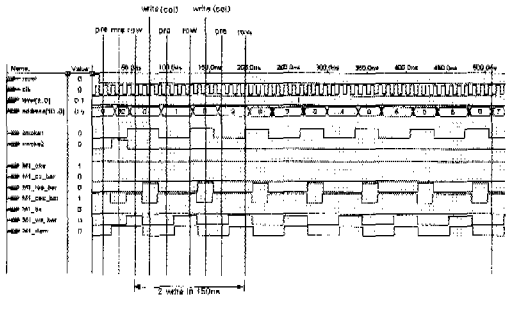


그림 14. 레벨 1의 열방향 DWT의 쓰기 동작 시뮬레이션 결과

VI. 결 론

본 논문에서는 이산 웨이블릿 변환(DWT)을 사용하여 2차원 영상압축기가 FPGA상에서 실시간 처리를 하도록 하는 외부 메모리 사상방법을 제안하였다.

E^2M^2(Efficient External Memory Mapping)이라 칭하는 이 메모리 사상방법은 DWT를 Mallat-tree방식으로 수행할 때 중간 결과를 저장하는 외부 메모리를 사용할 경우 메모리의 제약 조건과 연산 모듈의 데이터 처리방식의 상관관계를 모두 고려하여, 데이터 쓰기를 수행할 때는 메모리 셀의 행 방향을 고정시키고 열 방향의 주소를 1씩 증가시키면서 연속 2로 저장시키며, 데이터를 읽을 때는 반대로 열 방향을 고정시켜 놓고 행 방향을 연속 8로 읽는 방법이다.

Mallat-tree 원래의 이론대로 데이터를 저장시키면 데이터의 실시간 처리를 위해서 외부 메모리의 행 방향이 연속동작으로 읽기를 수행하여야 한다. 그러나 상용 SDRAM은 이러한 기능을 가지고 있지 않기 때문에 칩의 실시간으로 동작이 불가능하다. 이 경우 본 논문에서 제안한 E^2M^2방법을 C-언어를 사용한 에뮬레이터로 실험한 결과 원래의 Mallat tree이론을 그대로 적용했을 때와 마찬가지의 결과를 얻을 수 있었으며, 실제 100MHz로 동작하는 상용메모리를 선정하여, 메모리 제어기에 본 논문에서 제안한 E^2M^2방법을 적용시킨 결과, FPGA칩이 실시간으로 동작할 때, 칩과 메모리간의 필요한 접근 시간을 만족시켰다.

따라서, 영상을 압축하는 응용분야에서 본 논문이 제안한 메모리 사상방법을 적용하여 FPGA를 이용한 H/W를 구현할 경우 칩이 실시간으로 동작하므로, 실시간 영상처리를 위해 유용하게 사용될 수 있을 것으로 전망된다.

참 고 문 헌

[1] Gilbert Strang and Truong Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, pp. 337-401, 1997

[2] Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann, pp. 373-404, 2000

[3] J. D. Gibson,et al, *Digital Compressiong for Multimedia, Principles and Stansdards*, Morgan Kaufmann Pub., San Francisco CA, 1998.

[4] John G. Ackenhusen, *Real-time signal processing: Design and Implementation of signal processing systems*, Prentice Hall , pp. 290-319, 1999

[5] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation", IEEE Trans. on Pattern Anal.

Machine Intell., vol. 11, pp. 674-693, July 1989

[6] Colestock and M.A. "Wavelet-a new toll for signal processing analysts", IEEE 12th Digital Acoustics Systems Conf., pp. 54-59, Oct. 1993

[7] ISO/IEC JTC1/SC29/WG1,JPEG 2000 Editor Martin Boliek, Co-editors Charilaos Charilaos Christopoulos, and Eric Majani, *JPEG 2000 Part I Final Draft International Standard (Fromatted)*, 24 August 2000

[8] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression", IEEE Trans. on Image processing. vol. 9, no. 3, pp. 378-389, March 2000

[9] Ferretti, M.; Rizzo, D. "Handling borders in systolic architectures for the 1-D discrete wavelet transform for perfect reconstruction", IEEE Trans. on Signal Processing, vol. 48, issue 5, pp. 1365-1378, May 2000.

[10] Po-Cheng Wu, Liang-Gee Chen and Yeong-Kang Lai, "A block shifting method for reduction of block effects in subband/ wavelet image coding", IEEE Trans. on Consumer Electronics, vol. 44 issue 1, pp. 170-177, Feb. 1998.

[11] 공진흥, 김남영, 김동욱, 이재철, *VLSI 설계 이론과 실습*, 홍릉과학출판사, pp. 19-31, Jan 1998

[12] Ali M.Reza and Robert D., "FPGA implementation of 2D wavelet transform", Conference Record of the Thirty-Third Asilomar Conference on, vol. 1, pp. 584-588, 1999

[13] Chakrabarti, C. and Vishwanath, M., "Efficient realizations of the discrete and continuous wavelet transforms: from single chip implementations to mappings on SIMD array computers", IEEE Trans. on Signal Processing, vol. 43, issue 3, pp. 759-771, March 1995.

[14] M. Vishwanath, R. M. Owens and M. J. Irwin, "VLSI architectures for the discrete wavelet transform", IEEE Trans. on Circuits and Systems, vol. 42, No. 5, pp. 305-316, May 1995.

[15] Jijun Chen and M.A. Bayoumi, "A scalable systolic array architecture for 2-D discrete wavelet transforms", VLSI Signal Processing, vol. VIII, pp. 303 -322 , 1995

[16] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform", IEEE Trans. on Signal Processing, vol. 42, pp. 673-677, Mar. 1994.

[17] Ming-Hwa Sheu, Ming-Der Shi and Sheng-Wel Liu," A VLSI architecture design with lower hardware cost and less memory for separable 2-D discrete wavelet transform", ISCAS'98, vol. 5, pp. 457-460, 1998.

[18] 유지상, 김대경 외, "웨이블릿 기반 신호처리 시스템 설계 (응용분야)", 반도체 설계교육센터, 2000. 3. 22- 2000. 3. 24.

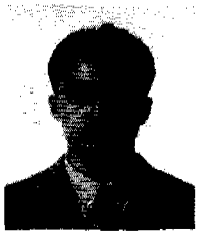
김 왕 현(Wang-hyun Kim) 준회원

2000년 2월 : 광운대학교 전자재료공학과 졸업.
2001년 3월~현재 : 광운대학교 전자재료공학과 석사과정.
2000년 3월~현재 : 인티스닷컴 (주) 연구원.
<주관심 분야> FPGA/ASIC 설계, DSP, Low power, Cryptosystem.

서 영 호(Young-ho Seo) 준회원

1999년 2월 : 광운대학교 전자재료공학과 졸업.
2001년 2월 : 광운대학교 전자재료공학과 대학원 졸업.
2000년 3월~현재 : 인티스닷컴 (주) 연구원.
2001년 3월~현재 : 광운대학교 전자재료공학과 박사과정.
<주관심 분야> FPGA/ASIC설계, DSP Cryptosystem.

김 종 현(Jong-hyun Kim) 준회원



1997년 2월 : 광운대학교 전자재료공학과 졸업.
1999년 2월 : 광운대학교 전자재료공학과 졸업.
2000년 3월~2001년 2월 : 인티스닷컴(주) 연구원.
2001년 2월~현재 : 광운대학교 전자재료공학과 박사과정.
2001년 3월~현재 : 펜타마이크로(주) 연구원.
<주관심 분야> VHDL, VLSI Testability, DSP, Design verification

김 동 욱(Dong-wook Kim) 정회원

1983년 2월 : 한양대학교 전자공학과 졸업.
1985년 2월 : 한양대학교 전자공학과 대학원 졸업.
1991년 9월 : Georgia 공과대학 전기공학과 졸업 (공학박사).
1992년 3월~현재 : 광운대학교 전자재료공학과 정교수. 광운대학교 신기술 연구소 연구원.
1997년 12월~현재 : 광운대학교 IDEC 운영위원.
2000년 3월~현재 : 인티스닷컴(주) 연구원.
<주관심 분야> 디지털 VLSI Testability, VLSI CAD, DSP 설계